UNIVERSIDAD CARLOS III DE MADRID

TESIS DOCTORAL

Open motion control architecture for humanoid robots

Autor:
Dmitry Kaynov

Director:
Carlos Balaguer Bernaldo de Quirós

DEPARTAMENTO DE SISTEMAS Y AUTOMATICA

Leganés, octubre 2008

TESIS DOCTORAL

Open motion control architecture for humanoid robots

Autor: Dmitry Kaynov

Director/es: Carlos Balaguer Bernaldo de Quirós

Firma del Tribunal Calificador:

		Firma
Presidente:		
Vocal:		
Vocal:		
Vocal:		
Secretario:		

Calificación:

Leganés de de

- To my family -

Acknowledgements

There are many people who helped me in producing this thesis. First of all, I would like to thank my supervisor, Prof. Carlos Balaguer, for having faith in me. Without the resources he provided, this research could have never happened. Without his guidance, this work would have never materialized and been completed.

I would like to thank all the members of the Rh-1 team. Without their work, I wouldn't have had such a capable platform to work with. Without the experiences we shared together, I would never have acquired the know-how to research the problems addressed in this thesis. I would particularly like to thank the primary team members who I have worked with over the years (in no particular order), Mario Arbulu, Luis Cabas, and Pavel Staroverov.

I extend my sincere gratitude to Dr. Jean Paul Leaumond and Dr. Eichi Yoshida, and all members of the GEPETTO research group from the LAAS-CNRS (French National Centre for Scientific Research) for their kind assistance with my work on the HRP-2 humanoid platform. I would especially like to thank Dr. Philippe Sourez for his help in the development of the stabilization control system without which this thesis could not have been completed.

I would also like to thank the European Commission, the Spanish government and the University Carlos III of Madrid for their financial support for the DPI2004-00325 (Development of a humanoid robot for cooperation with a human) and Robot@CWE (Advanced robotic systems in future collaborative working environments) projects from which this work was developed.

I would like to thank Kawada Industries, Inc. together with the Humanoid Research Group of the National Institute of Advanced Industrial Science and Technology (AIST) of Japan for building truly incredible robots and allowing us to work with them.

Further, I would like to thank all my colleagues from the Robotics Lab of the University Carlos III of Madrid, with whom it has been a great pleasure to work with during the past several years. I would like to thank all my friends, for their understanding and encouragement, as well as for providing much entertainment.

Finally, I would like to thank my family, especially my wife Elena and little Ivan for having faith in me finishing this work. They never doubted me for a minute and their confidence helped me in rough times.

Abstract

This Ph.D. thesis contributes to the development of control architecture for robots. It provides a complex study of a control systems design and makes a proposal for generalized open motion control architecture for humanoid robots.

Generally speaking, the development of humanoid robots is a very complex engineering and scientific task that requires new approaches in mechanical design, electronics, software engineering and control. First of all, taking into account all these considerations, this thesis tries to answer the question of why we need the development of such robots. Further, it provides a study of the evolution of humanoid robots, as well as an analysis of modern trends. A complex study of motion, that for humanoid robots, means first of all the biped locomotion is addressed. Requirements for the design of open motion control architecture are posed.

This work stresses the motion control algorithms for humanoid robots. The implementation of only servo control for some types of robots (especially for walking systems) is not sufficient. Even having stable motion pattern and well tuned joint control, a humanoid robot can fall down while walking. Therefore, these robots need the implementation of another, upper control loop which will provide the stabilization of their motion. This Ph.D. thesis proposes the study of a joint motion control problem and a new solution to walking stability problem for humanoids. A new original walking stabilization controller based on decoupled double inverted pendulum dynamical model is developed.

This Ph.D. thesis proposes novel motion control software and hardware architecture for humanoid robots. The main advantage of this architecture is that it was designed by an open systems approach allowing the development of high-quality humanoid robotics platforms that are technologically up-to-date.

The Rh-1 prototype of the humanoid robot was constructed and used as a test platform for implementing the concepts described in this Ph.D. thesis. Also, the implementation of walking stabilization control algorithms was made with OpenHRP platform and HRP-2 humanoid robot. The simulations and walking experiments showed favourable results not only in forward walking but also in turning and backwards walking gaits. It proved the applicability and reliability of designed open motion control architecture for humanoid robots.

Finally, it should be noted that this Ph.D. thesis considers the motion control system of a humanoid robot as a whole, stresses the entire concept-design-implementation chain and develops basic guidelines for the design of open motion control architecture that can be easily implemented in other biped platforms.

Keywords

Humanoid robots, control architecture, motion control, gait generation, walking stability, hardware architecture, software architecture, open systems.

Resumen

Esta tesis doctoral contribuye al desarrollo de arquitecturas de control para robots. Provee un estudio complejo de un diseño de sistemas de control y hace una propuesta para una arquitectura abierta generalizada de control del movimiento para robots humanoides.

En general, el desarrollo de robots humanoides es una labor científica y de ingeniería muy compleja que requiere nuevos enfoques en diseño mecánico, electrónica, ingeniería de software y control. En primer lugar, teniendo en cuenta estas consideraciones, esta tesis intenta responder a la cuestión de por qué necesitamos el desarrollo de tales robots. Además, proporciona un estudio de la evolución de los robots humanoides, tan bien como un análisis de las tendencias actuales. Posteriormente, se habla sobre un complejo estudio del movimiento que, para humanoides, implica ante todo la locomoción bípeda. Por último, se presentan los requisitos pare el diseño de una arquitectura abierta de control.

Este trabajo pone énfasis en algoritmos de control de movimiento para robots humanoides. La implementación de sólo servo-controles para algunos tipos de robots (especialmente sistemas andantes) no es suficiente. Incluso teniendo patrones estables de movimiento junto con un control bien ajustado, un robot humanoide puede caer mientras camina. Por lo tanta, estos robots necesitan la implementación de otro lazo de control superior que aportaría la estabilidad a su movimiento. Esta tesis doctoral estudia la problemática de un sistema de control de movimiento acoplado y propone una nueva solución al problema de la estabilidad caminando en humanoides. Se desarrolla un nuevo y original controlador de estabilización en el andar basado en el modelo dinámico del péndulo doble invertido desacoplado.

Esta tesis propone una novedosa arquitectura de control software y hardware de movimiento para robots humanoides. La principal ventaja de esta arquitectura es que fue diseñada para una metodología de sistemas abiertos, permitiendo el desarrollo de plataformas robóticas de alta calidad que son lo último tecnológicamente hablando.

El prototipo Rh-1 de robot humanoide fue construido y usado como una plataforma de test para la implementación de los conceptos descritos en esta tesis. También, la implementación de los algoritmos de control de la estabilidad andando fue hecha con la plataforma de OpenHRP y con el robot humanoide HRP-2. Las simulaciones y los experimentos del proceso de andar demostraron unos resultados favorables no sólo caminando hacia delante, sino también girando y retrocediendo. Ello demostró la aplicabilidad y la fiabilidad de la arquitectura abierta de control del movimiento diseñada para los robots humanoides.

Finalmente, hay que destacar que esta tesis considera el sistema de control de movimiento de un robot humanoide en su conjunto haciendo hincapié el proceso completo de conceptualización-diseño-implementación y desarrolla las líneas básicas para el diseño de la arquitectura abierta de control del movimiento que puede ser fácilmente implementado en plataformas bípedas

Contents

Chapter 1: Introduction **1**
1.1 The Motivation and Origin of the Thesis 1
1.2 Objectives of the Thesis 3
1.3 Contributions of the Thesis 4
1.4 Guide to the Thesis 4

Chapter **2: Trends in humanoid robotics** **7**
2.1 Introduction 7
2.2 General classification of robots by their motion ability 11
2.3 Why do we need to develop humanoids? 14
2.4 The history of humanoid research 17
2.5 Current humanoid research 34
2.5.1. Technological impact 35
2.5.2. Scientific (research) impact 35
2.5.3 Economical impact 36
2.6 Ethical concerns of humanoid research 37
2.7 Conclusion 39

Chapter 3: Concepts of the Robot's Control Architecture **41**
3.1 Introduction 41
3.2 System Architecture Definition 42
3.3 Control Architecture's Requirements 43
3.4 Design Levels of Control Architecture 47
3.4.1 Conceptual design. Basic Modes of operation in robotics systems 47
3.4.2 Functional design of control architecture 50
3.4.3 Physical design. Topology of a control system 53
3.5 The concept of a motion in robotics 56
3.6 Joint motion control 56
3.6.1 PID 59
3.6.2 PIV 62
3.6.3 Feedforward 62
3.6.4 Adaptive Control 64
3.7 Case study 66
3.7.1 Control architectures for polyarticulated robots 66
3.7.2 Control architectures for mobile robots 68
3.7.3 Control architecture of humanoid robots 69
3.7.4 Discussion 73

3.8 Motion control architecture of a humanoid robot 75
3.8.1 Conceptual level 75
3.8.2 Functional level 76
3.8.3 Physical level 78
3.9 Conclusions 80

Chapter 4: Joint control of a humanoid robot 83
4.1 Introduction 83
4.2 Joint Modelling 84
4.2.1. Complete dynamical model of a manipulator 85
4.2.2. Decoupled joint model 87
4.3. Identification design 90
4.4 Controller design 96
4.4.1. Theoretical Issues 96
4.4.2. Velocity Control 100
4.4.3. Position Control 101
4.5 Adaptive control 103
4.5.1. Load Torque 103
4.5.2. Moment of Inertia 106
4.5.3. Implementation 108
4.6 Conclusions 111

Chapter 5: Stabilization control of a humanoid robot 115
5.1 Introduction 115
5.2 Description of the humanoid robot walking 116
5.2.1 Bipedal locomotion characteristics 116
5.2.2 ZMP Concept 118
5.3 Biped walking dynamics modelling and patterns generation 124
5.3.1 Basic approaches 124
5.3.2 Inverted Pendulum Mode 126
5.3.3 Relation between COG and ZMP (ZMP equations) 130
5.3.4 Motion patterns generation 132
5.4 Stabilization Control 135
5.4.1 The need for stabilization control 135
5.4.2 ZMP control 137
5.4.3 Limits of the 3DLIPM and need of Attitude Control 139
5.5 Double Inverted Pendulum 142
5.5.1 Model design 142
5.5.2 LQR controller design 148
5.5.3 Decoupling the control 151
5.6 Stabilizer 154
5.6.1 General structure 154
5.6.2 ZMP controller design 155
5.6.3 Attitude controller design 161
5.6.4 Sensorial data acquisition and processing 164
5.6.4.1 ZMP measurements 165

5.6.4.2 ZMP data processing 170
5.6.4.3 Attitude measurements 171
5.6.4.4 Attitude data processing 173
5.6.5 Detailed Stabilizer architecture 182
5.7 Conclusions 183

Chapter 6: Open architecture for humanoid motion control 185
6.1 Introduction 185
6.2 Open architecture 186
6.2.1 Definition 186
6.2.2 Open software architecture 187
6.2.3 Open hardware architecture 188
6.2.4 Open and Standard 189
6.2.5 Degree of openness in the control architecture 190
6.2.6 Why open architecture for humanoid robot motion control?..... 192
6.3 Hardware architecture design 192
6.3.1 Considerations on hardware architecture design for humanoid robots 192
6.3.2 Basic systems of hardware control architecture 194
6.3.3 Hierarchy of the hardware architecture. Levels of control 196
6.3.4Main control system 197
6.3.4.1 Requirements 197
6.3.4.2 Embedded systems 197
6.3.4.3 CPU platform 198
6.3.4.4 Form factor 199
6.3.5 Joint control system 200
6.3.5.1 General scheme 200
6.3.5.2 Motor selection 201
6.3.5.3 Motion controller selection. Distributed vs. Centralized joint control 202
6.3.5.4 Joints synchronization in a complex multi-axis interaction 205
6.3.6 Communication system 205
6.3.6.1 Requirements 205
6.3.6.2 Proliferation of communication standards 206
6.3.7 Sensorial system 209
6.3.7.1 Requirements 209
6.3.7.2 Force/Torque sensor 209
6.3.7.3 Gyro sensor 210
6.3.7.4 Accelerometer sensor 211
6.3.8 Power control 212
6.3.8.1 Requirements 212
6.3.8.2 Power supply 212
6.3.8.3 Power conditioning 213
6.3.9 Proposed hardware control architecture scheme 214
6.4 Software architecture design 216
6.4.1 Considerations on software architecture design for humanoid

robots 216
6.4.2 Proposed software scheme 216
6.4.3 Control server 218
6.4.3.1 Requirements 218
6.4.3.2 Sample Time 220
6.4.3.3 State diagram 221
6.4.3.4 Control Area 222
6.4.3.5 Application Programming Interface 223
6.4.4 Control agent 225
6.4.4.1 Requirements 225
6.4.4.2 Motion interpolation 226
6.4.5 Server – Agent dynamical motion interaction 228
6.4.6 Client 232
6.4.6.1 Requirements 232
6.4.6.2 Humanoid Robot Supervisory Control System 233
6.4.6.3 HMI 235
6.4.7 Operating System 237
6.5 Communicational infrastructure design 238
6.5.1 Requirements 238
6.5.2 Organization and Control level communications 239
6.5.3 Device and Sensorial level communications 241
6.6 Conclusions 243

Chapter 7: Implementation of the motion control architecture with different humanoid platforms and walking experiments 245
7.1 Introduction 245
7.2 Humanoid Robot Rh-1 246
7.2.1 Considerations for the humanoid robot Rh-1 design 246
7.2.2 Mechanical design of Rh-1 247
7.2.3 Hardware architecture implementation 249
7.2.4 Software architecture implementation 251
7.2.5 Walking experiments 254
7.3 Humanoid robot HRP-2 262
7.3.1 Mechanical structure 262
7.3.2 Software environment 263
7.3.3 Stabilizer implementation 266
7.3.4 Simulation experiments 268
7.4 Conclusions 276

Chapter 8: Conclusions and future work 279
8.1 Conclusions 279
8.2 Future work 281

Appendix A: PC/104 standard 283
A.1 Form factor 283

A.2 Extension controllers 284

Appendix B: Communication technologies 285
B.1 Communication networks comparison 285
B.2 CAN bus 288
B.3 LAN/WLAN 290

Appendix C: CAN based communication protocols 293
C.1 CAN protocol 293
C.2 CANopen protocol 294

Appendix D: Real-time data transmission 297
D.1 Sensorial level Master-Slave real-time data transmission 297
D.2 Device level Client-Server real-time data transmission 298

Bibliography 301

Related Publications 309

List of Figures

2.1: Characteristics of industrial, service and personal robots 9
2.2: ABB industrial robots 11
2.3: Mobile robot a) PatrolBot mobile robot base b) MAGGIE mobile robot 12
2.4: Zoomorphic non-walking robots examples 12
2.5: Walking zoomorphic robots. a) Insect-like zoomorphic Phoenix hexapod robot
b) Animal-like zoomorphic robot AIBO 13
2.6: Android robots a) Walking humanoid robot ASIMO b) Repliee Q1 android 13
2.7: Hybrid robots example. a) ARMAR b) MANFRED 14
2.8: A humanoid robot in a domestic environment a) ASIMO b) Rh-1 15
2.9: a) HRP-1 driving a power shovel b) HRP-3 turning screws
c) Toyota Partner robot playing the violin 16
2.10: a) AMI 2003 b) Marie 2005 c) Toyota 2006 16
2.11: a) Steam Man b) Electric Man c) Boilerplate 18
2.12: Hardiman 18
2.13: a) WL-1 b) WL-3 19
2.14: a) WL-9DR b) WL-12RDIII c) WL-15R 20
2.15: a) WABOT-1 b) WABOT-2 21
2.16: a) WABIAN b) WABIAN-RII c) WABIAN-RIV d) WABIAN-RV 22
2.17: WABIAN -2R 22
2.18: E series a) E0, b) E1, c) E2 d) E3 23
2.19: E series a) E4, b) E5, c) E6 23
2.20: Prototypes a) P1, b) P2, c) P3 24
2.21: Honda ASIMO 25
2.22: H series a) H5, b) H6, c) H7 26
2.23: JOHNNIE 27
2.24: HRP humanoid a) HPR-2P b) HRP-3 27
2.25 Toyota Partner Robot 28
2.26: KHR humanoid a) KHR-2 b) HUBO (KHR-3) 29
2.27: Evolution of basic parameters of humanoid robots a) Height, b) Weight, c) D.O.F. 33
3.1: Task distribution between the user and the robot 44
3.2: Direct control scheme 48
3.3: Control scheme with teleoperation 48
3.4. Autonomous control scheme 49
3.5: Levels of the control architecture 50
3.6: Modularity level of control a) Integrated architecture b) Modular architecture 53
3.7: Centralization level of control. a) Centralized architecture b) Distributes architecture 54
3.8: Motion centered robotics system 57
3.9: Basic PID servo control topology 60

3.10: Change of response for varying a) K_p b) K_i c) K_d 61
3.11: Basic PIV servo control topology 62
3.12: Basic feedforward servo control topology 63
3.13: Gain scheduling adaptive control 64
3.14: MRAC adaptive control 65
3.15: Self Tuning adaptive control 65
3.16: Control architecture of ASIMO humanoid robot 70
3.17: Hardware control architecture of HRP2 humanoid robot 71
3.18: Robot Internal Network 72
3.19: Framework of a robot control architecture 74
3.20: Autonomous motion control of a humanoid robot 76
3.21: Functional hierarchy of a control system 77
3.22: General motion control architecture of a humanoid robot 78
3.23: Distributed hierarchical control architecture for humanoid robots 79
3.24: Framework of the Thesis 81
4.1: Joints' configuration of a humanoid robot 84
4.2: Typical joint of a humanoid robot 85
4.3: Scheme of a loaded joint 88
4.4: Block diagram of a joint model with DC motor 89
4.5: Motor 10 velocity output for the step current input (sample time 360 μs)
a) amplitude modifications b) frequency modifications 91
4.6: General-Linear Model Structure 91
4.7: Input-Output Model Structure a) ARX b) ARMAX c) OE d) Box-Jenkins 92
4.8: Model delay analysis. Motor 10 (sample time 360 μs) 93
4.9: Model order analysis. Motor 10 (sample time 360 μs) 94
4.10: Models analysis. Motor 10 (sample time 360 μs) 95
4.11: Root-locus of the estimated ARX220 transfer function of the motor 10 96
4.12: Cascade control scheme implemented in Rh-1 humanoid robot 97
4.13: Temporal output characteristics for the step input 98
4.14: Closed-loop system model 99
4.15: Velocity controller design (Z2=-25, K=0.02) 100
4.16: Experimental results for the motor 10 101
4.17: Experimental results for the motor 10. a) Real velocity/command velocity
b) position error 102
4.18: Different positions of the joint 10 in counts of the encoder a) 0 counts
b) -50000 counts c) -100000 counts d) -200000 counts 102
4.19: Joint 10 position error 103
4.20: Different positions of the link. a) PM=0º b) PM=15º c) PM=30º 105
4.21: Load torque influence experiments a) Velocity output and current input
b) Root-locus of the system 105
4.22: Simulation of influence of the moment of inertia a) Velocity output and current input
b) Root-locus of the system 107
4.23: Different positions of the link a) J max b) J med c) J min 108
4.24: Moment of inertia influence experiments a) Velocity output and current input
b) Root-locus of the system 108

4.25: Gain scheduling control algorithm 109
4.26: Velocity Command/Real a) Cascade Controller with constant parameters
b) Gain scheduling controller 110
4.27: Position Error a) Cascade Controller with constant parameters
b) Gain scheduling controller 110
5.1: Motion planes 117
5.2: Walking Cycle 118
5.3: Forces acting on the foot of the bipedal mechanism 119
5.4: M_{Ax} compensation 119
5.5: Forces acting on a humanoid (sagittal plane) 121
5.6: Humanoid robot a) Instable motion b) Stable motion 122
5.7: Stability region for a) Single Support Phase b) Double Support Phase 123
5.8: Pseudo-ZMP 124
5.9: Generalized mechanical structure of a humanoid robot 125
5.10: 3D Inverted Pendulum 126
5.11: 3D Inverted Pendulum with a contact polygon 130
5.12: 3D-LIPM with the ZMP in the origin point 133
5.13: COG trajectories with inicial conditions of the 3D-LIPM 134
5.14: Example of the motion pattern generated using 3D-LIPM 134
5.15: Stabilizer structure 136
5.16: Forces acting on the humanoid robot a) ZMP stable motion b) ZMP unstable motion ... 138
5.17: Forces acting on a foot 138
5.18: Humanoid robot ZMP error compensation 139
5.19: COG trajectory in the sagittal plane 140
5.20: Disturbing moment around COG 141
5.21: Humanoid robot motion modelling 142
5.22: Double inverted pendulum 143
5.23: Double inverted pendulum with upper part about zero equilibrium 146
5.24: Double inverted pendulum control system 149
5.25: Double inverted pendulum control $\vartheta_1(0) = 2$ deg, $\vartheta_2(0) = 1$ deg
a) ϑ_1 and ϑ_2 variation b) Control torques modification 150
5.26: Double inverted pendulum control $\vartheta_1(0) = -2$ deg, $\vartheta_2(0) = 0.5$ deg
a) ϑ_1 and ϑ_2 variation b) Control torques modification..... 151
5.27: Double Inverted Pendulum a) ϑ_2 variation b) ϑ_1 variation 152
5.28: Double Inverted Pendulum control. a)Influence of the ϑ_2 variation on ϑ_1.
b) Influence of the ϑ_1 variation on ϑ_2 152
5.29: a) ZMP Control b) Attitude control 154
5.30: Stabilizer controller structure 155
5.31: Inverted pendulum with compliant joint 156
5.32: ZMP LQR control system 159
5.33: ZMP Control. ZMP modifications 160
5.34: ZMP Control. Control signal modifications 160
5.35: Simple inverted pendulum 161
5.36: DC motor model 162

5.37: Attitude LQR controller 163
5.38: Attitude and control signal variation 164
5.39: ZMP measurement system. a) Mechanical configuration
b) Forces distribution diagram 165
5.40: Humanoid robot parts' dimension and distribution 167
5.41: Foot with mounted 6-axis force-torque sensor 168
5.42: Force-torque sensor implementation scheme a) Single support phase
b) Double support phase 169
5.43: Butterworth low-pass filter 170
5.44: Low-pass filter Bode diagram 170
5.45: Butterworth filter performance 171
5.46: Inertial reference system of the Upper body 171
5.47: Accelerometer measurements in sagittal plane a) static case b) dynamic case 172
5.48: An example of gyro's drift phenomenon 173
5.49: Attitude estimation system 174
5.50: Bode plot a) Accelerometer filter b) Gyro filter 176
5.51: Complementary filter performance 176
5.52: Kalman Filter Performance 181
5.53: Attitude estimation system 182
5.54: Stabilizer architecture 183
6.1: Openness of the control architecture for humanoid robots 191
6.2: Basic systems of a hardware control architecture 195
6.3: Hierarchy of hardware architecture 196
6.4: Joint control system 201
6.5: Centralized joint control 203
6.6: Distributed joint control 204
6.7: Communication standards within the realm of the industrial automation
communication hierarchy 207
6.8: Hierarchical Hardware architecture 214
6.9: Software architecture for humanoid robot motion control 217
6.10: Software architecture interaction 218
6.11: Control Server threads 219
6.12: Main computing and communication tasks of a humanoid robot 220
6.13: Server functioning state diagram 221
6.14: Control area modules 223
6.15: The link between user application and a humanoid robot 224
6.16: Motion program example 224
6.17: Control Agent threads 225
6.18: Motion interpolation 228
6.19: Server –Agents dynamic interaction 229
6.20: Control Server dynamical motion mode flowchart 230
6.21: Control Agent dynamical motion mode flowchart 231
6.22: HRSC Client Architecture 234
6.23: HMI modules 236
6.24 Ethernet based communication system 239
6.25: Encapsulation of data frame within a TCP datagram within an IP packet 240

6.26: The Application layer protocol package format 241
6.27: CAN bus based communication system 242
6.28: Reference model of a CAN based motion control system for humanoid robots 243
7.1: Mechanical configuration of the Rh-1 robot 247
7.2: Development of the Rh-1: a) Development in CAD/CAM and b) Final Prototype 248
7.3: Hardware distribution inside the humanoid robot 249
7.4: a) Knee joint cabling design b) Torso cabling design 250
7.5: Humanoid robot Rh-1 a) Cover design b) Realization 251
7.6: Control Server Terminal screenshot 252
7.7: HRSC client screenshot 253
7.8: One foot balancing experiment scheme 254
7.9: Rh-1 walking experiments scheme a) forward gait b) backward gait
c) lateral gait d) turning gait 255
7.10: Snapshots of forward walking gait 256
7.11: Joint position variation 257
7.12: Snapshots of the humanoid robot Rh-1 walking 259
7.13: Joint position variation (actual vs. target) 260
7.15: Joint current variation 261
7.16: HRP-2 humanoid robot and its joint location 263
7.17: OpenHRP software control system 264
7.18: The plugin manager structure 264
7.19: Screenshot of the Auditor 265
7.20: Stabilizer realization as plugins 267
7.21: HRP-2 angular motion 268
7.22: Hip joints and attitude variations a) Hip sagittal plane / Attitude Pitch
b) Hip frontal plane / Attitude Roll 269
7.23: Ankle joints and ZMP variations a) Ankle sagittal plane / x_{ZMP}
b) Ankle frontal plane / y_{ZMP} 269
7.24: Static test of the stabilizer in the sagittal plane a) attitude control b) ZMP control 270
7.25: Static test for Attitude control a) Hip control position and Ankle excitation variation
b) Pitch variation 271
7.26: Static test for ZMP control a) Ankle control position and Hip excitation variation
b) x_{ZMP} variation 271
7.27: Snapshots of humanoid simulator. Walking without stabilizer 272
7.28: Snapshots of humanoid simulator. Walking with developed stabilizer 273
7.29: Reference and real ZMP variation a) x_{ZMP} b) y_{ZMP} 274
7.30: Attitude variation 274
7.31: Real and reference joints variations. a) Right ankle sagittal plane
b) Left ankle sagittal plane c) Right ankle frontal plane d) Left ankle frontal plane
e) Right hip sagittal plane f) Left hip sagittal plane g) Right hip frontal plane
h) Left hip frontal plane 275
A.1: PC/104 a) form factor b) stack 283
B.1: NRZ compared with Manchester bit representation 289
B.2: Physical CAN bus connection 289
C.1: CAN message format 293
C.2: CANopen device realization for intelligent motion controller 295

List of Figures

D1: CAN protocol real time arbitration 297
D.2: CANopen dynamic buffer 298

List of Tables

2.1: Advantages and disadvantages of biped and wheeled robots 17
2.2: Comparative characteristics of the most significant humanoid robots:
WABIAN, WABIAN RIV, WABIAN 2 30
2.3: Comparative characteristics of the most significant humanoid robots:
P2, P3, ASIMO 31
2.4: Comparative characteristics of the most significant humanoid robots:
H6, H7, JOHNNIE 31
2.5: Comparative characteristics of the most significant humanoid robots:
HRP-2P, KHR-2, HUBO 32
3.1: Basic motion control characteristics of principal types of robots 73
4.1: Model delay analysis. Motor 10 (sample time 360 μs) 93
4.2: Model order analysis. Motor 10 (sample time 360 μs) 94
4.3: Model type analysis. Motor 10 (sample time 360 μs) 95
6.1: Communication performance of communication systems 208
6.2: Server Events 222
7.1: Specifications of the Rh-1 humanoid robot 248
7.2: Principal specifications of HRP-2 humanoid robot 262
B.1: Physical characteristics of communication standards 285
B.2: Transport mechanisms of communication standards 287

List of Tables

Chapter 1

Introduction

CHAPTER 1

Introduction

1.1 The Motivation and Origin of the Thesis

To design a machine that can duplicate the complexities of human motion has been a dream of humanity for decades .It was only approximately 15 years ago that the level of engineering (computing, mechanics, control, and materials) became sufficient to construct walking machines which could be called humanoid robots. Nowadays there are successful examples of this kind around the world, not only in Japan where research work started. The development of sophisticated humanoid robots has increased and it has become a very active area. Currently, there is growing interest not only in the academic field but in industrial fields as well.

The creation of a robot with human-like appearance manipulating other robots or machinery is a real possibility in the not so distant future. The enhanced locomotive ability of humanoid robots will increase the range of numerous manipulative functions. With the high level of integration of perceptive abilities, a humanoid robot will gain a certain level of autonomy by interacting with a person or executing different tasks intelligently. The inclusion of cognitive ability will make a humanoid robot capable of self-developing its mental and physical capabilities through real-time interaction with other robots, the environment and human beings.

The human-like locomotive ability of humanoid robots makes it possible to deploy them in places where humans are still working like machines, for example in jobs such as receptionist or worker on an automotive manufacturing line. Humanoid robots could also be developed to perform human tasks like personal assistance, where they should be able to help the sick and elderly. In the healthcare industry, humanoid robots could provide an invaluable service in the rehabilitation of patients with difficulties in motion capacities.

The humanoid robot is an ideal platform for tele-presence or tele-existence. Because of this, it is easy for us to predict its forthcoming applications in space exploration and police/military operations. At some point it should be possible to launch a space ship commanded by humanoid robots, yet controlled by humans at a ground station on Earth. It should also be possible to assign humanoid robots to operate military vehicles in a battlefield eliminating the risk of human casualty. In the current battle against terrorism, humanoid robots could help in prevent and rescue missions.

On the whole, as a humanoid robot is designed to replicate a human, it can use the same tools and operate the same equipment and vehicles designed for humans. Humanoid robots could

theoretically perform any task a human being can, so long as they have the proper software algorithms.

The implementation of humanoid robots with a high level of intelligence allows for the removal of some intelligence from the equipment. For example, a pilotless vehicle could be obtained using the usual vehicle with the humanoid robot inside. Furthermore, the same robot could be used to supervise a plant or to protect its owner. Nowadays the concept of universal robots is difficult to sell because the cost of labour is relatively cheap. This means robots are more likely to be used only where they can do a better job - precisely welding an automobile frame, for instance, or where humans can't or won't go, such as into a crippled nuclear reactor, or onto the surface of other planets.

Experts predict that the real boom in the use of humanoid robots will start by 2025 – 2030 when its fabrication and maintenance costs reduce enough to be profitable. There is no doubt that the emergence of humanoid robots will have a great impact on many aspects of our society.

A large number of research projects related to humanoid robots have been started recently. Different research lines are currently focusing on diverse aspects of control. Unfortunately, as the development of this type of robot is very complicated and expensive, and the level of competition is very high (an enormous empty market is waiting) information exchange between different groups is impossible. Hardware and software solutions are complex and closed. Although some control algorithms are well known and widely used, there is a need to organize and integrate different aspects of humanoid control as well as propose a unified model of control system for humanoid robot as a bipedal machine. In future, this model should be extended for the more general context of personal assistive robotics.

The current stage of development of control systems for humanoid robotics, which is a main theme of this thesis, is characterized by the apartness of the main elements composing the architecture. Research is mainly concentrated on the algorithms of the biped walking, without taking into account its complex relationship with other systems (mechanics, hardware, software) of humanoid robots. In this context a complex study that considers the motion control architecture of a humanoid robot as a whole is needed. It should stress a whole concept-design-implementation chain and propose some kind of methodology or guidelines for future developments in this field.

This thesis forms part of the Rh project (Development of a humanoid robot for cooperation with a human) launched by the Robotics Lab of the University Carlos III of Madrid. The main goal of this project is the development of a reduced weight human size robot which can be a reliable humanoid platform for implementing different control algorithms, human interaction, etc. Besides the control system design, the project includes a mechanical design, dynamics and kinematics studies and human-machine interaction design. The prototype of the humanoid robot was constructed and used as a test platform for implementing the concepts described in this Ph. D. thesis. Also, the implementation of walking stabilization control algorithms was made with the OpenHRP platform and the HRP-2 humanoid robot constructed by Kawada Industries Company.

1.2 Objectives of the Thesis

This Ph.D. thesis is centred on the control of the humanoid robot and attempts to discus problems and issues that should be considered when the control system of a humanoid robot is designed.

The principle objectives are:

- **To provide a study of historical perspectives and current research in the field of humanoid robots.** It is necessary to determine the impact of humanoids on our lives and development and study ethical concerns of humanoid research.
- **To study basic concepts of the control system of robots**. This means the study of general requirements, functional diversity and basic existing architectures. To classify existing robots by their main tasks and level of motion capacity. The next step is to focus on the architecture of humanoid robots describing their basic aspects as personal assistive autonomous robots with human-like appearance and bipedal locomotion.
- **To study control architectures of the most advanced humanoid robots** describing some aspects of their mechanical design, software, hardware and control solutions. Then, to highlight some advantages and disadvantages and establish current tendencies in humanoid robot design.
- **To propose novel open control architecture for humanoids**. The control architecture should be developed taking into account the functionality of a robot and its technical specifications. It is necessary to follow modern tendencies of architecture design in robotics. To highlight the advantages of the proposed architecture and make guidelines for its future development and improvements.
- **To develop new motion control algorithms.** Control architecture is set of hardware components and software algorithms. In order to establish the appropriate structure and links between the components of proposed architecture it is necessary to develop new control lows, such as joint control and walking stability control. These algorithms should provide fool functionality and reliability and improve already existing methods.
- **To implement the results in existing platforms and develop a new humanoid platform.** In order to prove the applicability of designed architecture it is necessary to implement it in a physical platform. The best way is to use commonly used platforms. Current work supposes the implementation of control algorithms with the HRP-2 humanoid robot constructed by Kawada Industries Inc. for Japan Intelligent Systems Research Institute (AIST) and now provided for the best research centres in Japan and Europe. The HRP-2 humanoid robot is considered by many specialists one of the most advanced contemporary humanoid platforms. The main problem with existing platforms is that the control architecture includes software and hardware components which can’t be transferred to other platforms without complete redesign. On the other hand, there are universal control algorithms allowing the implementation on multiple platforms. Therefore, it is necessary to develop a new humanoid platform using fresh principles for hardware/software design and in the same way propose a universal control algorithm capable of work in different existing hardware/software platforms of humanoid robots. The humanoid robot Rh-1 which is under construction in the

Robotics Lab of University Carlos III of Madrid is considered the best platform to implement this kind of architecture.

1.3 Contributions of the Thesis

It should be mentioned that this thesis tries to propose a new complex approach to the design of a control system for humanoid robots. All aspects of the design process as well as detailed architecture and control algorithms were developed and implemented.

The main contributions of this thesis focus on the following:

- **Development of open motion control architecture for humanoid robots.** This architecture comprises all aspects of a humanoid robot motion interaction with the key point being a bipedal locomotion. We demonstrate the utility of open principles in control architecture design. This allows a more flexible and adaptable system with the capability to change its properties for the user's needs. Proposed hardware architecture is a novel solution in the area of humanoid robots which complies with modern tendencies in robotics. Open software architecture providing the robot with a standard functionality is easily upgraded facilitating new functionality.
- **Development of new efficient algorithms for motion control and stabilization of the humanoid robot walk.** These algorithms provide simple solutions allowing fast and reliable integral control of a robot. New methods of processing sensorial data needed for stabilization are proposed. Through experiments realized with different humanoid platforms it is demonstrated that proposed control can be used to improve robot performance.
- **Implementation of the proposed software and hardware architecture with the humanoid robot Rh-1.** As the development of a robotics platform from the "zero" base is a very complex task, the obtained prototype Rh-1 capable of making steps, is a successful experimental result of this thesis. It provides the open knowledge data of "know-how" for the next generation of humanoid robots. In general, the review of the fundamental aspects connected with the design and implementation of the control architecture finally, can lead to the establishment of a series of criterion facilitating the design process of the humanoid robot from the control point of view. It will establish some kind of methodology which permits an adaptation of functional requirements and existing technical solutions. Also, it allows us to unify the development of humanoid robot prototypes in terms of costs as well as functionality.

Finally, it should be mentioned that the proposed thesis is a complex study that considers the motion control system of humanoid robots as a whole. It stresses a whole concept-design-implementation chain and tries to propose original and efficient solutions at all levels.

1.4 Guide to the Thesis

The material of this thesis is arranged into VIII chapters as follows:

- All readers are recommended to start with chapter 2 for background information and a history of the humanoid systems which are the main topic of this thesis. In this chapter the

author addresses the question of why we need to develop a humanoid robot. Chapter2 also describes the great impact of humanoid development on many areas of our life. Finally it considers some ethical concerns around humanoid research.

- Chapter 3 studies the basic concepts of robot control architecture. It defines robot control architecture as a system and poses requirements for its design. Furthermore, it stresses the concept of motion in robotics and provides discussion on the motion control architectures of the most significant contemporary robots. Finally in this chapter the conceptual design of the motion control architecture for humanoid robots is provided.
- Chapter 4 focuses on the joint control for humanoid robots. The modelling and identification of joints in humanoid robot is discussed. Moreover, the chapter provides the possible design of joint motion controllers. It takes into account the influence of the continuous change of load torques and moments of inertia on the controllers' behaviour. The experimental results obtained are shown and discussed.
- Chapter 5 provides the design of the stabilization control algorithms for humanoid robots. Initially, a short introduction on the theory of biped locomotion is presented. Later, humanoid dynamics and the stable walking patterns generation methods are considered and requirements for stabilization control posed. Chapter 5 shows the modelling and design of the stabilization control system under double inverted pendulum dynamics. Further studies are provided on the dynamical model of a robot to simplify the control by decoupling the dynamics. In addition, this chapter presents and develops new methods for sensorial data acquisition and processing which are indispensable for practical implementation of the system. Finally, it provides the design and implementation of the stabilizer architecture.
- Chapter 6 addresses the design of open architecture for humanoid motion control. It defines the open software and hardware systems in the framework of humanoid robot control. This chapter provides the requirements and detailed development of hierarchical modular hardware architecture. Also, it provides the development of client – server software architecture as well as communication infrastructure for humanoid robots.
- Chapter 7 presents the implementation of the motion control architecture with different humanoid platforms. The experimental results are obtained in walking experiments with Rh-1 humanoid robots and simulation experiments with the OpenHRP platform of the HRP-2 humanoid robot.
- Chapter 8 summarizes the thesis and proposes considerations for future work in this field.
- Appendix A studies the PC/104 computer standard suggested for the implementation of the hardware architecture for humanoid robots.
- Appendix B considers basic communication technologies implemented in this thesis.
- Appendix C shows detailed description of implemented CAN and CANopen protocols.
- Appendix D describes the realization of real-time mechanisms in Sensorial and Device levels of motion control architecture.

Chapter 2

Trends in humanoid robotics

CHAPTER 2

Trends in humanoid robotics

2.1 Introduction

The concept of artificial intelligence and artificial humans has never failed to excite scientists, engineers, writers and movie producers. It is the dream of many scientists and engineers to build artificial humans that will one day be part of our society. The creation of an artificial being is also thought to be the pinnacle of humankind's scientific and engineering achievement. This, at present, has only appeared in the realms of fantasy and science fiction and engineers and scientists have made only small steps forward in this growing field in the past few decades. Therefore a research work on the humanoid robot is one of the hottest topics in engineering research today, driven by the desire to create fully autonomous humanoid robots that can think and move around like human beings.

The word "Robot" first appeared in Karel Capek's 1921 play - *"Rossum's Universal Robots" (R.U.R.)* where the *Robots* were human-like machines made to replace human workers. It comes from the Czech word "Robota" which means "labour doing compulsory manual works without receiving any remuneration" or "to make things manually".

Robots are now very widely used in the manufacturing sector. Robotic technology has been developed and refined so successfully that an entire manufacturing process can be handled by robots alone. On the other hand, robot designs have evolved such that the main consideration is the tasks to be achieved, not the appearance. Thus, most robots that we see today do not resemble humans. They are simply working machines meant to maximize efficiency.

The International standard ISO 8373 defines a "Robot" as: "An automatically controlled, reprogrammable, multi-purpose, manipulator, programmable in three or more axes, which may be either fixed in place or mobile for use in industrial automation applications".

This definition involves the concept of a manipulator but restricts the area to only one type of contemporaneous robot - an industrial manipulator. By including the perception of the environment and a capacity for action with some level of autonomy the robot "leaves" the manufacturing plant. The continuous evolution of robots needs a more general definition to include other types of robots in the global robotics area. The Oxford dictionary defines "Robot" as "a machine resembling a human being and able to replicate certain human movements and functions automatically." Nowadays, the robot is leaving factories and laboratories and slowly entering society in the form of a service robot.

Other types of robots have been developed which are determined as service robots. Progress in the field of electronics, especially in sensors, actuators and control algorithm has opened up new areas for robots. In this context, a possible definition of a service robot was suggested by the

Fraunhofer Institute for Produktionstechnik und Automatisierung (IPA) [Schraft 94]: "A service robot is a freely programmable cinematic device which performs services semi- or fully automatically. Services are tasks which do not contribute to the industrial manufacturing of goods but are the execution of useful work for humans and equipment. The actual execution of tasks by the service robot can be a series of complex movements, which can also be carried out when influenced by unforeseeable occurrences or environmental conditions. A service robot must therefore be able to act, within certain limits, independently." Here, the service robot can be understood as a robot able to operate autonomously and execute different tasks (not only manufacturing) for the benefit of a humans. It should be mentioned that the service robot can be used in manufacturing plants and the industrial robot or manipulator can be used for non-productive tasks. There are a many examples of industrial robots or manipulators of a general use programmed to carry out service tasks without having specially adapted design.

Industrial robots are usually designed taking into account different requirements for velocity, load capacity, accessibility etc. and always have an anthropomorphic design as a manipulator, with a different number of degrees of freedom (basic criterion for this design will be discussed in following chapters). Industrial manipulators don't have any problems with weight distribution because they usually have a fixed permanent place in a production line where heavy components can be easily located.

Service robots differ from industrial robots because they should be individually designed for the execution of a given task, taking place in a specific environment, following a predefined organizational scheme. The successful design of service robots should be based on detailed knowledge of available technologies and methodologies for design handling devices or mobile platforms, peripheral devices, and organizational schemes which account for a flexible, fault-tolerant and user friendly human-machine interaction. These robots have a wide field of applications which means the need for very high flexibility of operations to adapt the robot to different situations it may face during execution of a service task. They should operate in non-structured environments modelled for human beings and not be suited specifically to a concrete task. Also, as a human can be located in the same environment, there is the problem of safe interaction between robot and human to consider. This manifests a good number of technical, as well as some ethical, problems. To overcome these problems some safety requirements for service robot design have been proposed, which can be unified as a "Human-friendly robot" concept. It addresses the human-robot interaction, executing different tasks and guarantees safety in every moment. Further development of the concept of a robot leads to the next, more advanced step of evolution in robotics – a personal robot.

Like the personal computer, the personal robot is one that will change the use of robots from being large, expensive, and hard to use, to being small, inexpensive, and easy to use. Turning a conventional service robot into a universally applicable personal robot or artificial servant is mainly a programming task. The improvements in motion planning, computer vision (especially scene recognition), natural language processing, and automated reasoning are indispensable to make this a possibility. Figure 2.1 below presents the basic and most representative characteristics of industrial, service and personal robots as a function of degree of their autonomy and flexibility of the environment where they operate.

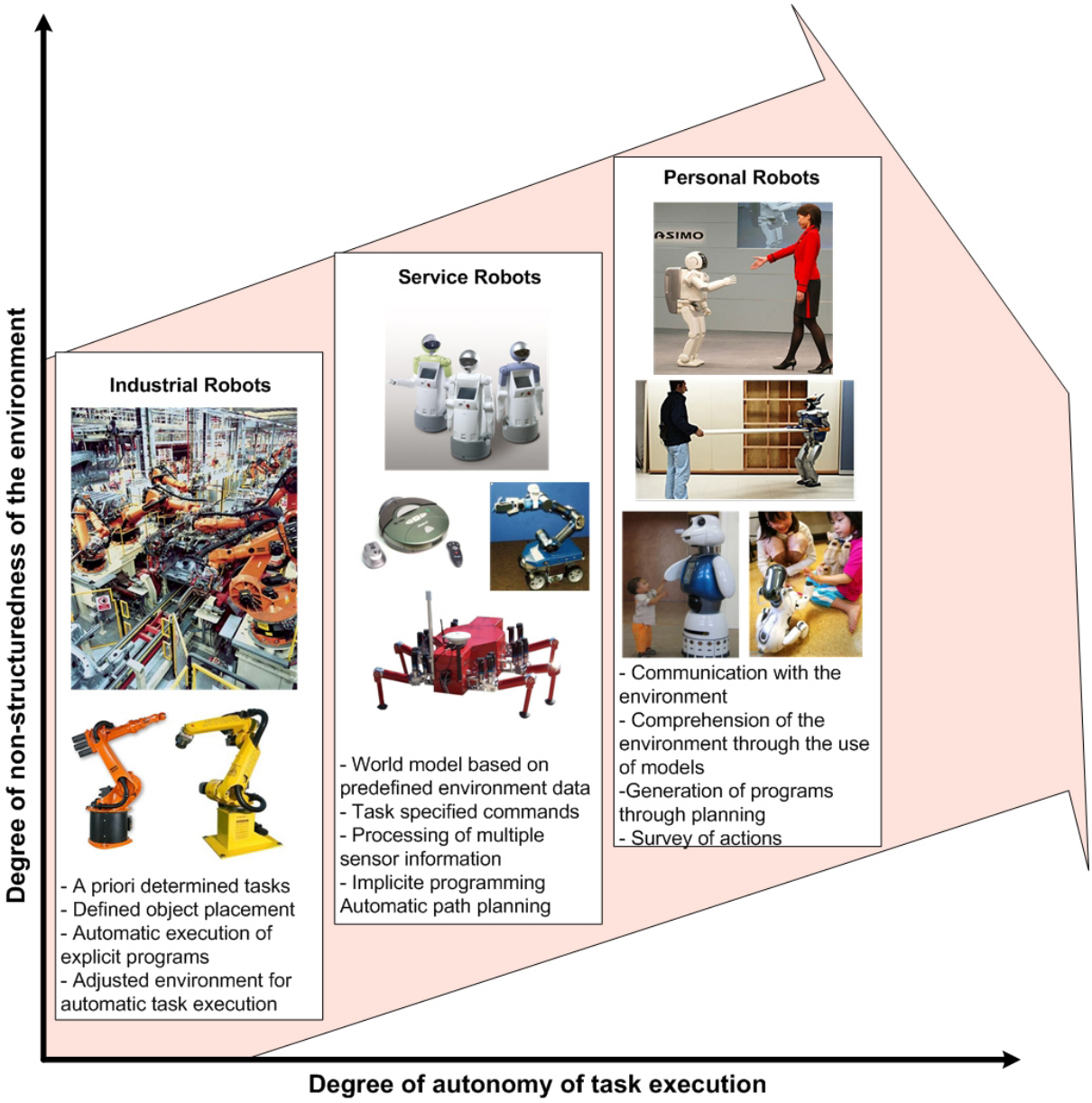


Fig. 2.1: Characteristics of industrial, service and personal robots.

In this context, a humanoid robot – the main topic of this Ph. D. thesis can be considered as the modification of service and personal robots allowing them to manifest human–like appearance. Following sections set out to prove that the humanoid robot can be considered evolutionary peak of robotics systems from mechanical, hardware and software and control points of view.

The free web encyclopedia Wikipedia defines a “humanoid robot” as a “robot with its overall appearance based on that of the human body”. A more precise definition would seem to be the following one: “A humanoid robot is the embodiment of manipulative, locomotive, perceptive, communicative and cognitive abilities in an artificial body similar to that of a human, which

possesses the skills to execute motions with a certain degree of autonomy, and can be advantageously deployed as an agent to perform tasks in various environments.“ [Xie 03]

In general, a humanoid robot is a robot which has a torso, head, two arms and two legs, although some forms of humanoid robots may model only part of the body, for example, from the waist up. Some humanoid robots may also have a “face”, with “eyes” and “mouth”. The principal traits of humanoid robots are a resemblance to human beings and of course bipedal locomotion. In addition, humanoid robots can be divided into Androids - humanoid robots built to resemble a male human, and Gynoids - humanoid robots built to resemble a human female.

Humanoid robots can be considered as autonomous robots because they can adapt to changes in their environment or themselves and continue to reach their goal. This is the main difference between humanoids and industrial robots. While industrial robots are used to performing tasks in highly structured environments, humanoids can be used in our normal everyday environment. In this context, some of the capacities of humanoid robots should include, among others:

- Safe interaction with human beings and the environment
- Autonomous learning, which means that a humanoid should be able to learn or gain new capabilities without outside assistance, adjust strategies based on the surroundings and adapt to new situations it may face
- Some level of self maintenance including the ability to recharge itself, swap batteries, etc.

Since humanoid robots are created to imitate some of the same physical and mental tasks that humans execute every day, scientists and specialists from many different fields including engineering, cognitive science, and linguistics must combine their efforts to create a robot as human-like as possible. The main goal of all creators of humanoid robots is that one day they will be able to understand human beings and reason and act like we do. If some day humanoids were able to do so, they could eventually work alongside humans and even replace humans in some circumstances. Another important benefit of developing humanoids is the increased understanding of the human body’s biological and mental processes, from the simple (for us) act of walking, to the complex concepts of consciousness and spirituality.

Like other mechanical robots, humanoids have basic components of their architecture like: Sensing, Perception, Actuating, Planning, Decision-Making and Control. Since they try to simulate human structure and behaviour and they are autonomous systems, most of the time humanoid robots should be more complex than other kinds of robots. Concerning the locomotion side of the interaction, for example, a mobile robot, which is another type of autonomous system, is less complex in this respect.

The complexity of a humanoid robot affects all robotic scales (mechanical, spatial, time, system and computational complexity), but it is more noticeable in mechanical and control system complexity scales. In the first place, current humanoids aren’t strong enough even to jump and because the power/weight ratio is not as good as in the human body. The appearance of new light materials and other actuators instead of an electrical motor may change this situation in the future. On the other hand, there are very good algorithms for the several areas of humanoid control, but it is very difficult to merge all of them into one efficient system because the overall system’s complexity becomes very high. Nowadays, these are the main difficulties that humanoid robot development has to deal with.

2.2 General classification of robots by their motion ability

As it will be demonstrated later, most contemporary robotic systems can be classified based on their ability to make different types of motions. Due to this, their control architectures differ radically. Five basic groups of robotic systems can be distinguished by their motion control architecture: Polyarticulated, Mobile, Zoomorphic, Anthropomorphic and Hybrid robots.

Polyarticulated robots - This group contains different forms and configurations of robots with one common characteristic – stationarity. These robots are structured in order to move terminal effectors in the determined working environment in one or several systems of coordinates, and with a limited number of degrees of freedom (Fig. 2.2). Although, exceptions may exist when a robot is guided in space (with moving platform) in order to perform a task in another environment. This group of architectures includes all manipulators and industrial robots. These robots are used when it is necessary to attend rather extensive but permanent working zones, working mainly with different types of objects and environments. They are now in service in virtually all branches of the motor industry and its suppliers, as well as in medical, education and training applications. With their powerful control architectures they provide cost-effective, reliable and easily-installable solutions for everything from simple tool and component handling tasks, to complex applications in which the entire system is controlled by the robot. [Tanner 81]

Fig. 2.2: ABB industrial robots.

Mobile robots – This group has a great motion capacity due to implemented wheel based cart or platform systems (Fig. 2.3(a)). They mainly execute different telecontrolled tasks and/or are driven by the information received from the integrated sensorial system. The motorized turtle designed in 1948 by Walter was the first predecessor. From the beginning of the sixties mobile robots were designed and implemented within industry. These robots were able to transport parts from one point of the production line to another, guided by preplanned paths materialized by the electromagnetic or photoelectric bands from circuits mounted into the floor. From the beginning of the seventies a lot of work was related to major autonomy of mobile robots. It involved providing the mobile robot with a vision system [Moravec, 1981]. Finally, from the beginning of the eighties, when more complex and precise sensorial systems appeared, the development of architectures for control of mobile robots was concentrated on the superficial intelligence and decision making systems [Bares, 1998], [Thorpe, 1990]. Mobile robots provided with this kind

of control system are usually able to plan motions and avoid obstacles. Today, research is also centred on human-mobile robot interaction [Khamis 2007] figure 2.3(b).

a) 

b) 

Fig. 2.3: Mobile robot a) PatrolBot mobile robot base b) MAGGIE mobile robot

Zoomorphic robots – This type of robot is characterized by the locomotion system which imitates the locomotion of diverse living beings. Although there can be a lot of morphological differences between all variations of zoomorphic systems, it is possible to distinguish two basic categories: walking and non-walking zoomorphic architectures.

Non-walking Zoomorphic robots – this group of zoomorphic robots currently is not very well evolved. One example of this is the modular snake-like robot Polybot [YIM 2000] (Fig. 2.4). It is also necessary to mention robots based on the two tank-like caterpillar treads, able to climb stairs and negotiate uneven surfaces [Zhang 2006].

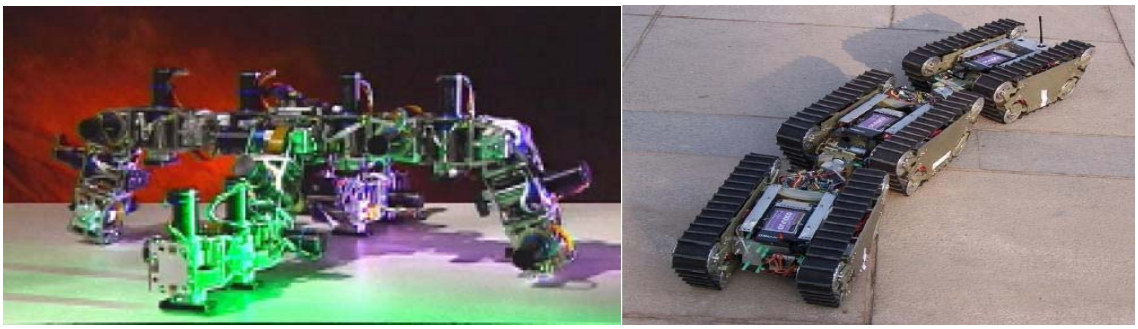


Fig. 2.4: Zoomorphic non-walking robots examples

Walking Zoomorphic robots – this type of robot is very well presented. There are a lot of laboratories developing walking zoomorphic architectures able to work in every kind of terrain. The two main types of research can be categorised into: insect-like robots and animal-like robots. Insect-like robots usually are biologically inspired [Quinn 98] and have from 4 to 12 controllable legs and insect like bodies (Fig. 2.5(a)). It allows them to copy the motion of insects in order to

achieve very good stability and cross-country ability. They have a really wide range of applications. It could be spatial research, out-of-the-way terrain exploration, or volcanic research. Animal-like robots try to imitate the movements of animals and are usually constructed for research and entertainment [Hornby 2000] (Fig. 2.5(b)) tasks. The control of this kind of robot is more complicated than control of a mobile or polyarticalated robot because of the need to maintain the equilibrium at every stage of motion.

a)  b) 

Fig. 2.5: Walking zoomorphic robots. a) Insect-like zoomorphic Phoenix hexapod robot b) Animal-like zoomorphic robot AIBO

Anthropomorphic robots (Androids) - or a humanoid (bipedal) robot, as it was mentioned above, try to reproduce the body and behaviours of a human. Presently the research on humanoids is increasing rapidly, although, there still remains a lot of work ahead. One of the basic challenges in this field is to reproduce human-like motion abilities beginning with the bipedal locomotion (Fig. 2.6(a)) [Hirai 98].

a)  b) 

Fig. 2.6: Android robots a) Walking humanoid robot ASIMO b) Repliee Q1 android

The motion control architecture in this case is the most complex compared with the other robot types presented above. The main challenge is being able to control and coordinate in real time

the dynamics of the entire body and maintain the equilibrium in the single support phase, i.e. when the robot is supported only by one foot. The control architecture of this kind of robot is an aim of the presented research and will be discussed and developed further in the following chapters.

Another complex aspect related to androids is the ability to reproduce the human upper body, especially the face (Fig. 2.6(b)) [MacDorman 2005]. The difference between a humanoid robot and android is only skin-deep. The latter looks exactly like a human on the outside, but internally has the mechanics of a humanoid robot.

Hybrid robots – this type of robot architecture corresponds to robotic systems difficult to classify in either group presented above. Their structure combines properties of various types of robot. For example a human-like body with a wheel base has at the same time attributes of mobile and anthropomorphic robots [DILLMANN 04] (Fig. 2.7(b)). Another example is the system, a combination of the wheelbase and a manipulator similar to the industrial robot [Blanco 2005] (Fig. 2.7(b)). Located in the same category are robots which can't be considered as androids or mobiles robots. They include for example, personal robots.

a) 

b) 

Fig. 2.7: Hybrid robots example. a) ARMAR b) MANFRED

In the following chapter all presented architectures (except the hybrid architecture because it is a compound produced by the combining of others) will be studied and compared from the motion control point of view in order to make some guidelines for designing the open control architecture for humanoid robots.

2.3 Why do we need to develop humanoids?

The main peculiarities of humanoid robots compared to other kinds of robots such as industrial manipulators, mobile robots, multi-legged robots can be posed as [Balaguer 2006]:

- A human body-like mechanical structure with two legs, two arms, a torso and a head. This leads to the high complexity of the mechanical system and a hyper DOF system with more than 20 joints. Therefore, the dynamics and kinematics of a humanoid robot are more complex and require more computational resources.

- Bipedal human-like locomotion which is characterized by a stable walking gait. As a humanoid robot changes its position using only two legs, the dynamics of the system vary. There are two different dynamical situations during a step – single support and double support phases. This leads to the implementation of extremely complex systems for maintaining the stable walking of a robot.
- As humanoid robots are considered to be autonomous, they should be provided with on-board power and computer autonomy. This requires implementation of complex real-time control architecture.

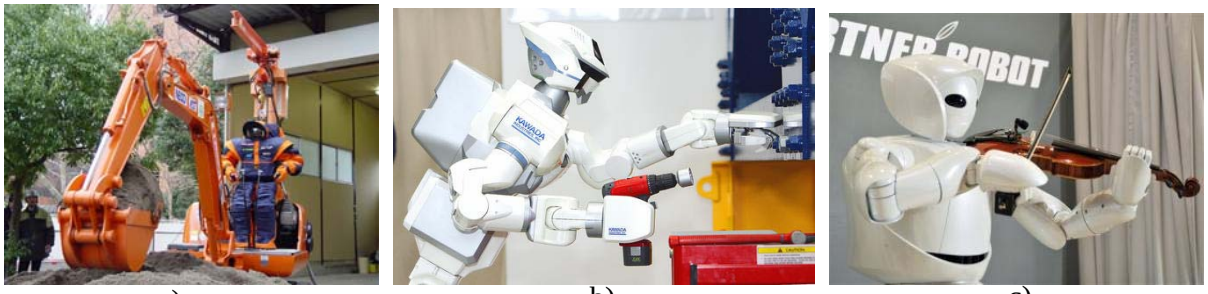
Thus, generally speaking, the development of humanoid robots is a very complex engineering task that requires new approaches in mechanical design, electronics, software engineering and control. So, taking into account all these considerations, why do we need the development of such robots?

In order to answer this question we need to take a look at our every day environment. Our homes, streets and transportation systems are adapted for us (human beings). The same occurs when we look at the tools we usually use. So the right question is: Why do we need to adapt everything to robots when we can adapt a robot for us? While other types of robots designed to execute a specific task need to be adapted for drastic changes in their environment, humanoid robots can work directly in the same environment as humans without any modification (figure 2.8).

a)  b) 

Fig. 2.8: A humanoid robot in a domestic environment a) ASIMO b) Rh-1

Moreover, because they have the same anthropomorphic structure they can use the same tools and even the same machines that are designed for humans (figure 2.9).

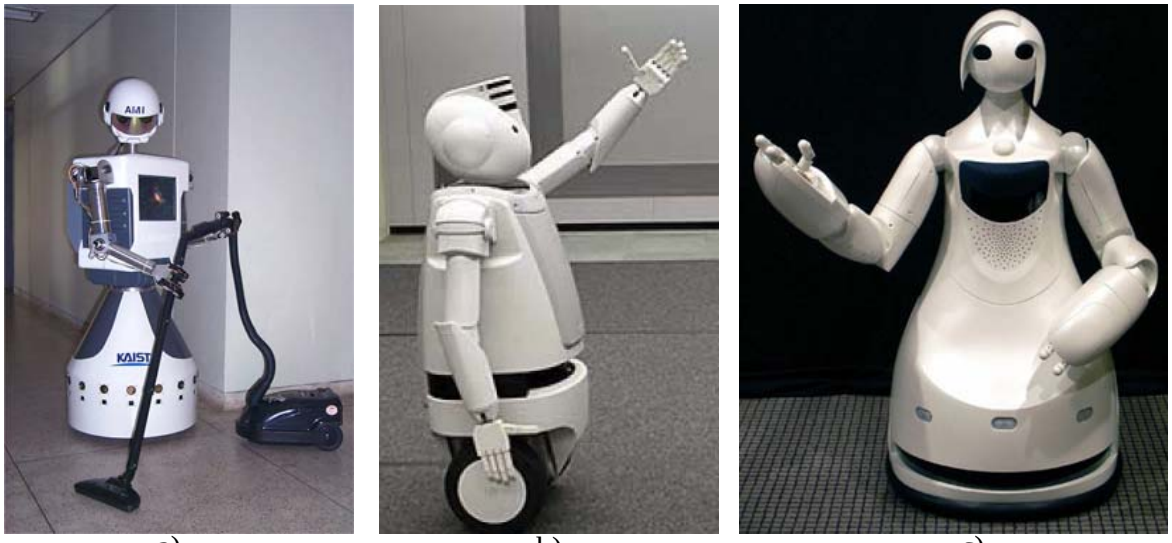


a) b) c)

Fig. 2.9: a) HRP-1 driving a power shovel b) HRP-3 turning screws c) Toyota Partner robot playing the violin

Another reason for the use of humanoid robots lies in the area of human psychology. It is easier for a human to interact with a human-like being. Humanoid robots provide the possibility of comfortable interaction with humans using the same and easy comprehensible gestures and movements.

Another question posed by the developing of humanoid robots is their motion abilities. It is clear that wheeled locomotion can sometimes be more effective than the biped one. Therefore, a lot of wheeled humanoid robots have been developed recently (figure 2.10).



a) b) c)

Fig. 2.10: a) AMI 2003 b) Marie 2005 c) Toyota 2006

However, the use of wheeled humanoid robots is very restricted by the requirement of a plain terrain which is not always possible. Our daily environment contains a lot of different ramps and stairs which are an insurmountable obstacle for wheeled robots. The following table presents some evident advantages and disadvantages of biped and wheeled humanoid robots.

	Advantages	Disadvantages
Wheeled robots	Easy locomotion	Need to adapt the environment
	Easy design	Difficulties in manoeuvres
	High speed locomotion	Only flat terrain
	Easy control	
Biped robots	No need to adapt the environment	Complex locomotion
	Natural human-like appearance	Complex design
	Any terrain is suitable	Low speed of locomotion
		Complex motion control

Table 2.1: Advantages and disadvantages of biped and wheeled robots

Drawing on this comparison we can conclude that if the environment is “empty” and the floor is flat, the wheeled humanoid is the best solution. If the environment is real (stairs, not flat floor, cluttered and without modification) the biped humanoid is the best choice.

The biped robot a very complex system but it has other advantages that lie in the field of intelligence and cognition. In the case of humans, bipedism liberated our hands to create tools and start cognition. Thus, the achieving of a stable biped locomotion for a humanoid robot could be considered the first stage in the creation of an intelligent robotics system. Therefore nowadays, work on humanoid robots is focusing more on bipedism than cognition.

Currently there are a considerable number of problems in humanoid robotics to solve:

- Stable bipedal locomotion is not totally achieved; we have only some good lab examples
- There are no human-size humanoids presently on the market and probably will not be in the near future
- It is necessary to have robust biped platforms in order to implement cognitive robotics

Thus, comparing the current state of humanoid robotics with human evolution it could be said that we are in the pre-robotic stage of development. .

Finally, it should be pointed out that the last and maybe most powerful reason for developing and promoting humanoid robots in every day life, is the dream of generations to have a human-like creature being our servant and friend. This leads to some ethical concerns regarding the development of humanoid robots, which will be discussed in following sections.

2.4 The history of humanoid research

In this section, the historical and current research works related to humanoid robots will be considered. Brief descriptions of these works will be included. Due to the great number of works, only significant full size humanoid and bipedal robots will be considered.

The history of the development of humanoid robots is curiously long and detailed. Leonardo da Vinci designed and possibly built the first humanoid robot. The robot was designed to sit up, wave its arms, and move its head via a flexible neck while opening and closing its jaw.

Other designs, related to the appearance of steam power and electricity appeared in the 19th century. They were simple automates which were able to imitate some restricted human and animal movements. John Brainerd created the Steam Man apparently used to pull things (Fig. 2.11(a)). Frank Reade Jr. described the Electric Man which is more-or-less an electric version of the Steam Man (Fig. 2.11 (b)). Dr. Achibald Campion built the Boilerplate which is a prototype soldier (Fig. 2.11 (c)).

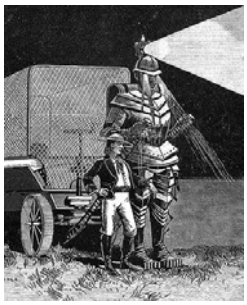
a)  b)  c) 

Fig. 2.11: a) Steam Man b) Electric Man c) Boilerplate

The first bipedal machine was constructed by George Moore in 1893 but real progress was achieved only at the end of the 1960s when the electronics, mechanics and materials became sufficiently advanced to create such complex a system as a bipedal robot. Firstly, the research was centred on the design of an artificially powered exoskeleton to increase the physical strength of a man. These systems were called man-amplifiers. The General Electric company was the pioneer in this field and in the middle of the 1960s presented the first exoskeleton - Hardiman (fig. 2.12) [GE 66].

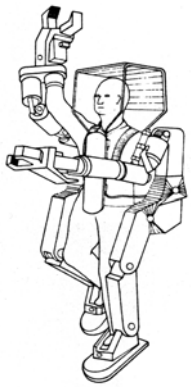


Fig. 2.12: Hardiman

However, as the servo control was not developed enough, this machine was practically useless. All these systems were very unstable and it was very difficult to maintain their equilibrium. Later on, the research line moved to the design of machines which were able to walk automatically, thus beginning the current era of the humanoid robot.

It is necessary to mention the great impact of the research work of M.Vukobratovic on the progress in humanoid walking stability. Vukobratovic and his colleagues from the Belgrade Mikhail Pupin Institute proposed a theoretical model to explain and control biped locomotion at the Third All-Union Congress of Theoretical and Applied Mechanics in Moscow in January 1968. The fundamental concept of his model is called the Zero Moment Point (ZMP) [Vukobratovic 69]. Zero Moment Point is a concept related to dynamics and control of legged locomotion, e.g., for humanoid robots. It specifies the point at which dynamic reaction force on the contact of the foot with the ground does not produce any moment, i.e. the point where total inertia force equals 0 (zero). The zero moment point is a very important concept in the motion planning for biped robots. Since they have only two points of contact with the floor and they are supposed to walk, “run” or “jump” (in the motion context), their motion has to be planned concerning the dynamical stability of their whole body. This is not an easy task, especially because the upper body of the robot (torso) has a larger mass and inertia than the legs which are supposed to support and move the robot. Proposed concepts for the control of a humanoid robot, which remains a central point, will be discussed more in detail in following chapters.

In order to simplify the control, thirst bipedal walking robots were made without taking into account the whole body dynamics. A bipedal robot has the same bottom part (legs) but does not have body, arms and head. The ability to walk is more critical for the humanoid robot than for a biped one, although the design process is rather similar. Also a lot of research centres started to design arms and heads separately in order to add them later to a humanoid robot.

The Japan University of Waseda has played a fundamental role in the evolution of humanoid robots. In the middle of 1960s this university launched different research projects lead by Prof. Ichiro Kato. As a result, since 1967, the series of WL robots has appeared.

The WL-1 robot (Fig. 2.13(a)) was the first one in this series, which culminated in the WL-16R bipedal robot in 2004 [Sugahara 2004]. The WL-1 robot which was capable of making a few steps and the successor, WL-3 (Fig. 2.13(b)) was able to sit down and stand up using its electro hydraulic actuators.

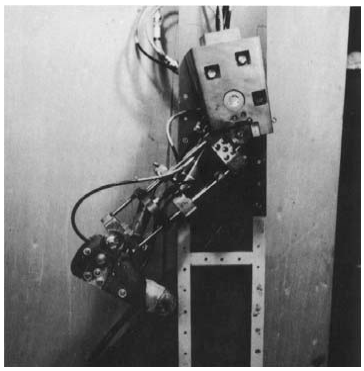
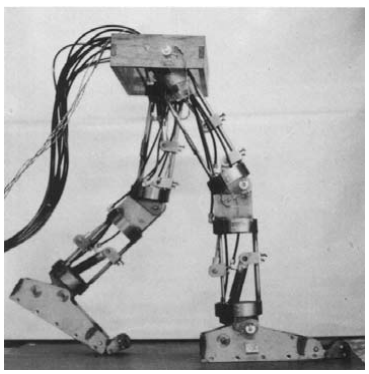
a)  b) 

Fig. 2.13: a) WL-1 b) WL-3

The development of this series continued during the 1980s, introducing improvements in control and mechanical systems of robots. The WL-9DR (Fig. 2.14(a)) constructed in 1980 was the first bipedal robot imitating a human walking almost to perfection.

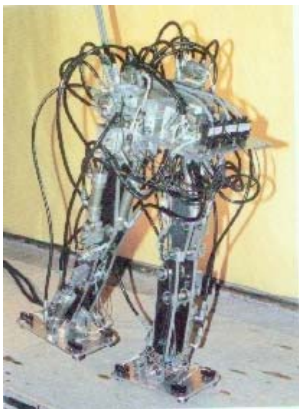
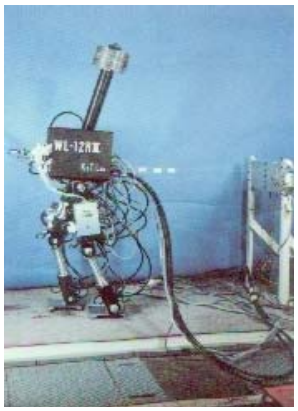
a)  b)  c) 

Fig. 2.14: a) WL-9DR b) WL-12RDIII c) WL-15R

In 1982 and 1984 new improvements appeared in the form of the WL-10R and WL10RD prototypes. In the mechanical structure design of these robots some new materials (light plastics) were implemented in order to decrease robot weight and simplify the control. Also, additional degrees of freedom were added to allow robots to walk straight forward, backward and realize some lateral displacements.

The WL series was continued with the WL-12 and WL-12RDIII (Fig. 2.14(b)) prototypes in 1986. The last one was able to operate in a human environment, that is, it was able to go up and downstairs, walk stably and avoid some obstacles. Its movements were extremely similar to a human. The walking adaptive control implemented in these bipeds was very advanced for that time and provided the system with a high level of stability. Finally, the WL series culminated in the WL-16R biped (Fig. 2.14(c)) adopting as legs a pair of 6-DOF parallel mechanisms [Hashimoto 2005]. The application of WL-15R is intended for us in the fields of welfare, as a walk support machine or a bipedal vehicle as an alternative to wheelchairs, which can go up and down stairs.

Applying the experience acquired in the WL bipedal series development, the University of Waseda developed WABOT-1 (WAseda robot) (Fig. 2.15(a)), the first constructed bipedal robot with humanoid appearance (with complete torso) in the world. The WL-5 biped was adopted as its artificial legs.

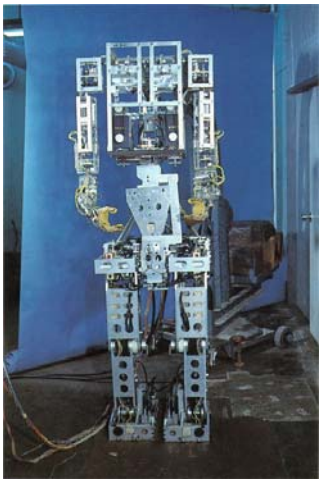
a)  b) 

Fig. 2.15: a) WABOT-1 b) WABOT-2

The robot consisted of a limb-control system, a vision system and a conversation system. The conversation system allowed it to maintain a conversation in Japanese, and a sensory system allowed it to measure distances and directions to objects. The WABOT-1 walked with its lower limbs and was able to grip and transport objects with its hands, which used tactile-sensors. It is estimated that the WABOT-1 had the mental faculty of a one-and-half-year-old child.

The second prototype WABOT-2 (Fig. 2.15(b)) was developed between 1980 and 1984. It was defined as a "specialist robot" rather than a versatile robot like the WABOT-1. The robot musician WABOT-2 can converse with a person, read a normal musical score with its eyes and play tunes of average difficulty on an electronic organ. The WABOT-2 is also capable of accompanying a person while they sing. The WABOT-2 can be cited as one of the first milestones in the development of a "personal robot".

The family of humanoid robots constructed in the University of Waseda continued with another series known as WABIAN (WAseda BIpedal humANoid) (Fig. 2.16(a)) [Lim 2000]. The development of this series started in 1995 and it continues to the present. The main design goal was to obtain a system capable of interacting with a human. Thus the height of the robot was selected as the average height of Japanese people. The robot had 5 D.O.F. with electrical servo motors inside each joint and walking velocity comparable with a human. The computational system was integrated inside the body, however the power supply source was external and it limited the autonomy of the robot. This robot was able not only walk forwards and backwards but also to carry out more complex movements like dancing. Also it was able to manipulate (transport) light objects and interact with humans.

The subsequent prototypes (Fig. 2.16(b), 2.16(c), 2.16(d)), WABIAN-R (1996), WABIAN-RII (1996), WABIAN-RIII (1997), WABIAN-RIV and WABIAN-RV were constructed as improvements on the previous ones. These robots demonstrated advances such as the expansion of degrees of freedom in the mechanical structure, better stability and bigger weight of objects to be transported.

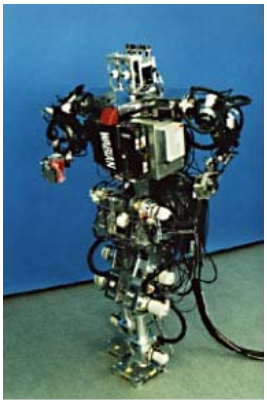
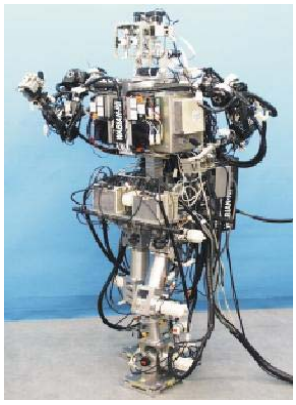
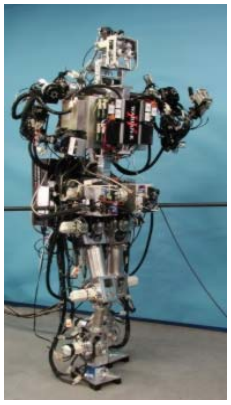
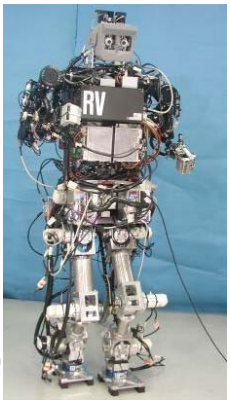
a)  b)  c)  d) 

Fig. 2.16: a) WABIAN b) WABIAN-RII c) WABIAN-RIV d) WABIAN-RV

The WABIAN series is still in development. Current research work is concentrated on the WABIAN 2 humanoid robot (Fig. 2.17) [Ogura 2006]. Implementing contemporary technologies and materials the new version of the robot has became one of the best in the field. The main advantage of the mechanical design is the use of 2-DOF (Roll, Yaw) in the waist area which enables more human-like walking motions. This new mechanism has an advantage which allows the robot to walk with a knee stretched position, due to the independent orientation of trunk movement. WABIAN-2R, the last robot of the series, was developed in order to mimic various motions which humans commonly produce. Link length and movable range are designed to facilitate human motion measurement for rehabilitation.

a)  b)  c) 

Fig. 2.17: WABIAN -2R

The development of humanoid robots is a very active area not only in academic circles but also in industry. The Honda Motor Company, for example, is another great centre of research on humanoids. In the mid 1980s, Honda engineers set out to create a walking robot. During the first 7 years of their work they developed 7 bipedal robots denominated from E0 up to E6.

Early models (E0, E1, E2, E3) was focused on developing legs that could simulate the walk of a human. E0 (Fig. 2.18(a)) was the first in a series of successive bipedal humanoid models created by Honda. Created in 1986 the robot walks on two feet, in a manner resembling human

locomotion, taking 5 to 20 seconds to complete a forward step. It could only walk in straight lines and perform “static walking” which vastly differs from how humans walk. In static walking, both feet start on the ground together. As the next step is taken, the robot’s centre of gravity gradually moves to one foot, until it supports the whole body weight before taking the next step. Between 1987 and 1991 the company constructed the E1, E2 and E3 (Fig. 2.18(b), 2.18(c), 2.18(d)) prototypes with the ability to walk more like a human. This was called “dynamic walking”. In dynamic walking when people take a step forward, they get just off balance, as if they are about to fall forward. Right before losing balance, people throw another foot ahead to support the body. Then, this series of movements repeated, becomes a walk.

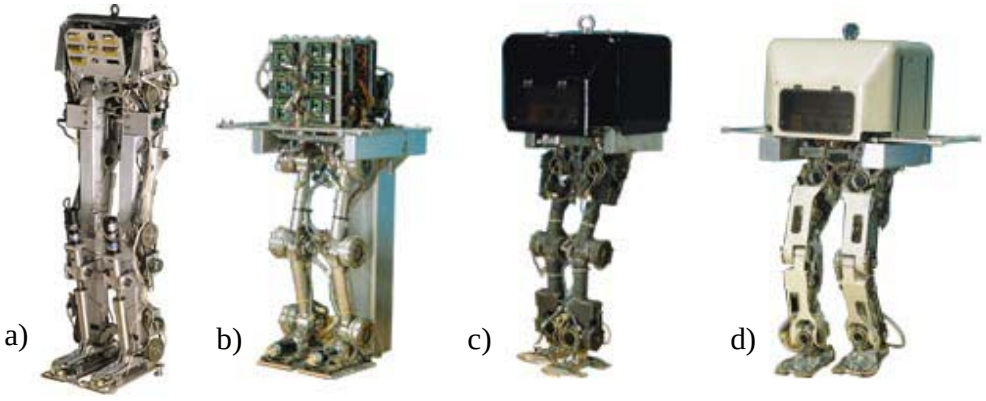


Fig. 2.18: E series a) E0, b) E1, c) E2 d) E3

The next series of models, E4, E5 and E6 (Fig. 2.19) were focused on walk stabilization and stair climbing. These prototypes were more sure-footed bipeds capable of working faster, climbing stairs and walking up and down an incline. With control of leg movement firmly established, it was a time to move to the next challenge – attaching the body, functional arms and a head. Thus, these bipeds became the precursors of other series, by now, really humanoid robots.

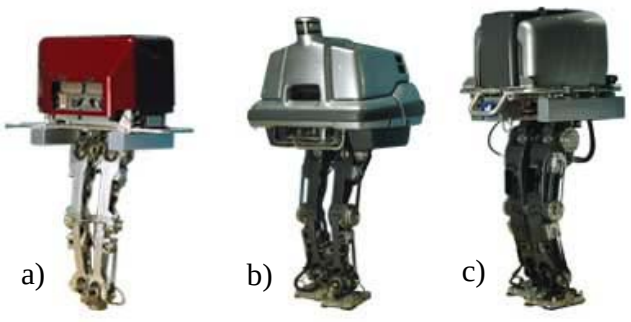


Fig. 2.19: E series a) E4, b) E5, c) E6

In 1993 Honda presented the P1 (Fig. 2.20(a)) humanoid robot. It was based on the E6 bipedal platform with upper limbs and a body. The robot was rather rugged at 1950 mm tall and

weighing 175 kg. The robot could turn external electrical and computer switches on and off, grab doorknobs and pick up and carry things. Research was also carried out on coordination between arm and leg movements.

The P2 robot, launched in 1996 (Fig. 2.20(b)) improved upon the previous robot with a more friendly design, improved walking, stair climbing/descending, and wireless automatic movements. The robot could walk smoothly over uneven surfaces and bumps and even maintain balance when firmly pushed while standing still. It had a height of 1820 mm and weighed 210 kg. Using wireless technologies, the torso contained a computer dedicated to controlling the robot's movements, motor drives, battery, wireless radio and other devices, all of which were built in. Independent walking, walking up and down stairs, cart pushing and other operations were achieved without wires, allowing independent operation.

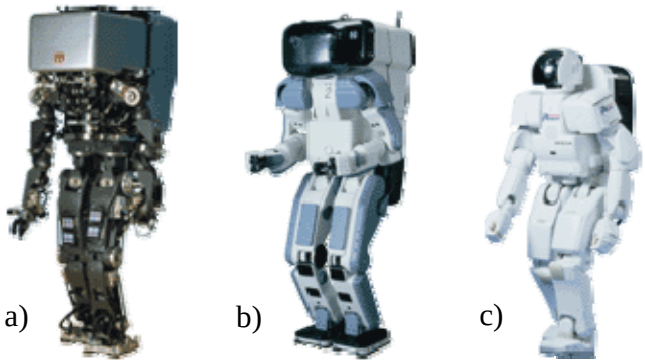


Fig. 2.20: Prototypes a) P1, b) P2, c) P3

The P3 (Fig. 2.20(c)) model, presented in1997, was even more compact, standing 1600 mm tall and weighing 130 kg. Size and weight were reduced by changing component materials and by decentralizing the control system. Its smaller size was better suited for use in the human environment [Hirai 98]. The P3 humanoid robot was able to move faster with better balance and smoother motion.

Undoubtedly, the ASIMO (Advanced Step in Innovative MObility) (Fig. 2.21) is the culmination of two decades of humanoid robotics research by Honda's engineers. ASIMO can run, walk on uneven slopes and surfaces, turn smoothly, climb stairs, and reach for and grasp objects. ASIMO can also comprehend and respond to simple voice commands. ASIMO has the ability to recognize the face of a select group of individuals. Using its camera eyes, ASIMO can map its environment and register stationary objects [Sakagami 2002]. The robot can also avoid moving obstacles as it moves through its environment. After more than 15 years of research and development, the first version of ASIMO appeared in 2000 and was the world's most advanced humanoid robot. Standing at 1300 mm and weighing 54 kg, the robot resembles a small astronaut wearing a backpack and can walk in a manner similar to human locomotion at up to 6 km/h.

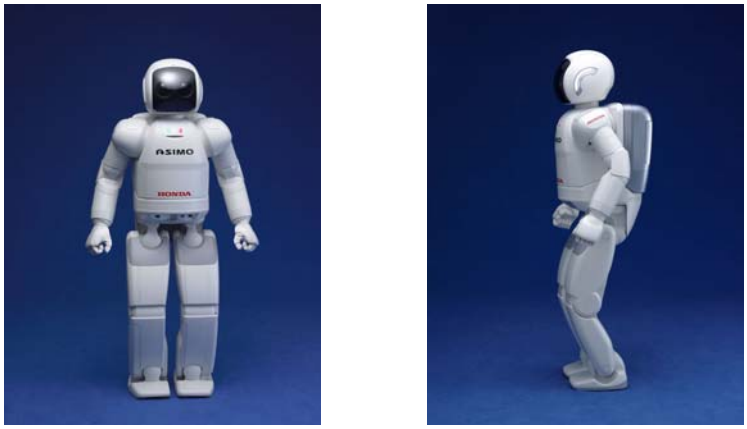


Fig. 2.21: Honda ASIMO

ASIMO overcame the main problem of its predecessors – the huge size and awkwardness which impede their movement in the human environment. ASIMO's height was an important strategic decision. Its eyes were located at the same level as a seated adult. In terms of walking abilities, ASIMO achieved some breakthroughs. If early robots had to stop to make sharp turns, ASIMO with its sophisticated hip joints had the ability to move and turn smoothly and execute walking patterns, coming even closer to human movements. In December 2005, the latest version of ASIMO was introduced. It featured more fluid and dynamic movement made possible through an expansion of its degrees of freedom from 26 to 34. A major advancement with the new prototype is its ability to change direction while running.

The Jouhou System Kougaku laboratory (JSK) from the University of Tokyo also played an important part in the development of humanoid robots. The construction of the prototypes was realized in cooperation with the Aircraft and Mechanical Systems Division of Kawada Industries, Inc. The research line adopted in this project was called "Remote Brained Robotics" based on placing the artificial intelligence (the brain) of the robot outside the body [Kanehiro 99]. This allows the use of external and very powerful computers which are too big to be placed inside the humanoid robot body. The brain and the body of the robot are connected via wireless communications. This approach has some advantages. It allows for the sharing of the development environment by different prototypes. As well as the growing requirements of the brain, software can be adapted to the actual hardware evolution of the external control PC without modifications inside the robot's body. Of the robots constructed using this design concept it is necessary to distinguish the H5, H6 and H7 prototypes (Fig. 2.22). The first H5 (1999) prototype stood at 1.27 m and weighed 33 kg, with 30 DOF.

a) 

b) 
c) 

Fig. 2.22: H series a) H5, b) H6, c) H7

The following model, H6 was constructed in 2000 and was 10 cm taller, weighing 55 kg with 35 DOF. The robot had an artificial vision system allowing it to generate real time motion pattern to move easily in complex environments [Kagami 2000]. The last prototype, H7 presented in 2001 was 1.47 m tall with a weight of 58 kg and 30 DOF [Kagami 2001].

Although Japan is the country where research on bipeds and humanoids is most advanced, in other countries some impressive humanoid robots have also been constructed. There were some significant works realized by Massachusetts Institute of Technology (MIT), USA, for example. The research started in the mid 1990s with the design of bipedal robots such as the Spring Turkey [Pratt 96], M2 [Robinson 99] and Troody (from 1996).

In Europe the research on humanoid robots started much later than in Japan or the USA. The technological advances are therefore more modest than in these two countries. Nevertheless, it is important to mention the humanoid project JOHNNIE (Fig. 2.23) realized by the German University of Munich. JOHNNIE's structure resembles the human locomotor apparatus and has a total of 17 actively driven joints [Gienger 2000]. The overall robot weight is about 49kg, the height is 1,80m. The robot is able to walk on even and uneven ground and around curves. Further, a jogging motion is planned, which is characterized by short ballistic phases where neither foot is in contact with the ground.

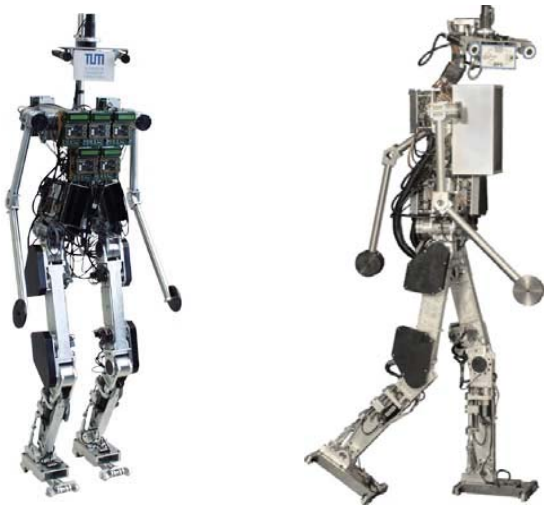


Fig. 2.23: JOHNNIE

The robot is autonomous in terms of actuators, sensors and computational power, while the energy is supplied by a cable. The robot can walk stably and some external disturbances are eliminated effectively. Today, Johnnie can walk with a speed of 2.2km/h.

One the most advanced humanoid robots is the HRP-2P (Fig. 2.24) constructed by Kawada Industries [Kaneko 2004]. It is taller (1,54 m) but lighter (58 kg) than Honda's Asimo and has 30 degrees of freedom (Fig. 2.24(a)).

a) b)

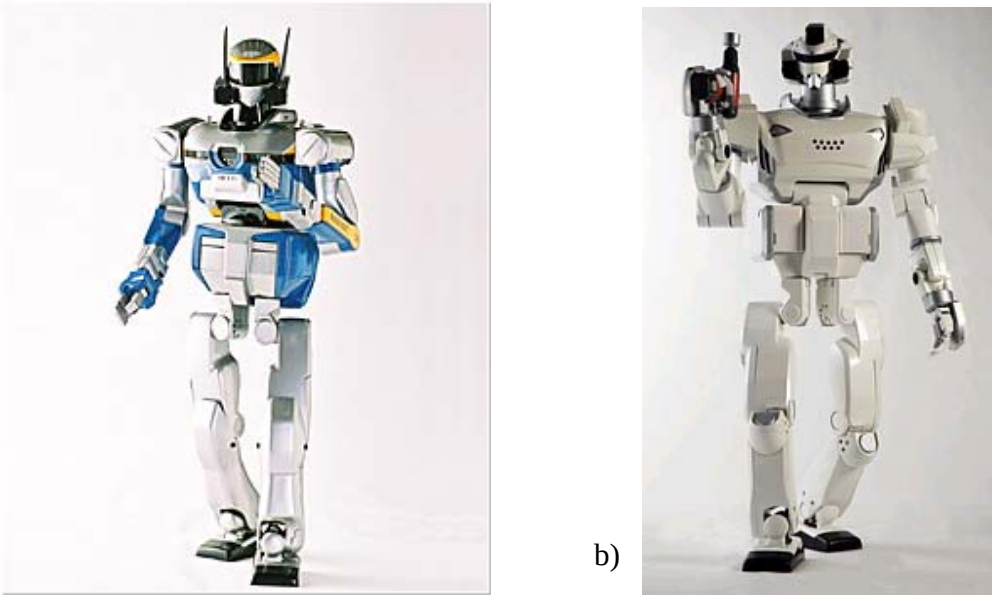


Fig. 2.24: HRP humanoid a) HPR-2P b) HRP-3

The distinguished feature of HRP-2P is its light weight and the removal of a backpack which the conventional humanoid robots have. The software platform, called OpenHRP, is a collection

of software for humanoid robots including a dynamics simulator and controllers. HRP-2P and OpenHRP are research platforms for humanoid robotics. The HRP-2P is the first humanoid robot that is the size of a human and can lie down and get up. Biped walking is realized by a walking pattern generator using a preview control and a feedback control. The lying down and getting up motions are made possible by HRP-2P's compact body with a waist joint and a novel motion controller. This robot is being used to explore future robot technologies with respect to walking on uneven surfaces, tipping-over and falling control mechanisms, as well as, “human-interactive operations in open spaces”. Subsequently Kawada Industries presented the last version of the HRP-3 humanoid robot (Fig. 2.24(b)) with improved locomotion and capability to operate different tools [Akachi 2005].

In 2004, the Japanese company Toyota started its project “Partner Robots” designed to function as personal assistants. One of the robots in this series is a humanoid robot (Fig. 2.25). Partner robots was designed to have humanoid characteristics, such as being agile, warm and kind and also intelligent enough to skilfully operate a variety of devices in the areas of personal assistance, care for the elderly, manufacturing, and mobility.

This model walks on two legs similar to a person, making it easy to become accustomed to. The humanoid’s height is 120 cm and weight 35 kg. Through the expanded development of the driving control technologies for auto mobiles, Toyota came up with new stabilizing technologies for robots. Small, light-weight and low-cost high-precision-sensors were developed based upon automotive sensor technology, and are used as an attitude sensor that detects a tilt of a robot. It is able to use its hands to carry out a wide variety of tasks. In order to enable the robots to play musical instruments, Toyota developed artificial lips that it claims move with the same finesse as human lips, which, together with robots' hands, enable the robots to play trumpets like humans do.

Fig. 2.25 Toyota Partner Robot

It is able to use its hands to carry out a wide variety of tasks. In order to enable the robots to play musical instruments, Toyota developed artificial lips that it claims move with the same finesse as human lips, which, together with robots' hands, enable the robots to play trumpets like humans do.

Another considerable project realized by the Korea Advanced Institute of Science and Technology (KAIST). KHR-1 (KAIST Humanoid Robot platform-1) was started in 2002 [Kim 2002]. The total weight, including batteries, computer, controllers and amplifiers, is 48 kg and its height is about 120 cm. The KHR-1 had 21 Degrees of freedom. By experimenting with this prototype, the validity of the actuator selection method was verified.

KHR-2 (Fig. 2.26(a)) is the succeeding mode of KHR-1. Its development started in 2003, and its platform was finished in 2005. Compared with KHR-1, it has almost twice as many joints (41 DOF), and better operating system from DOS to Windows. In the sensor field, inertia sensors, tilt sensors and CCD cameras were added with F/T sensors. Walking algorithm also had been upgraded following studies, providing the robot with more stable walking [Park 2005].

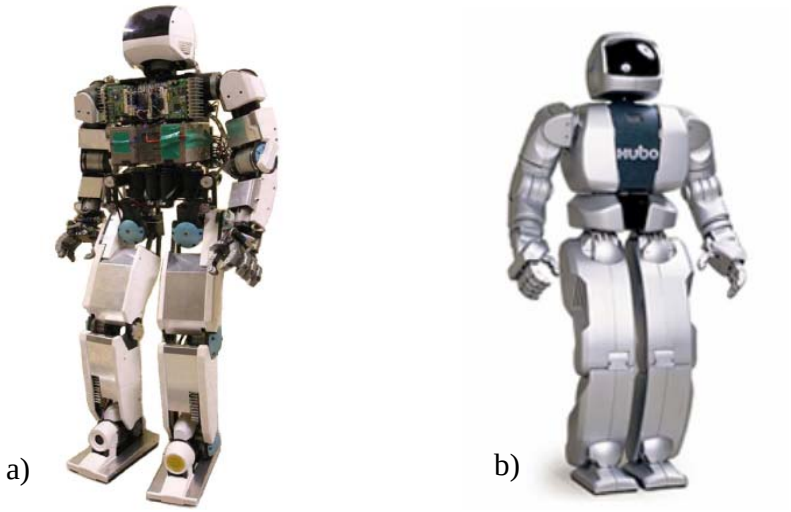


Fig. 2.26: KHR humanoid a) KHR-2 b) HUBO (KHR-3)

Hubo (Fig. 2.26(b)), developing name KHR-3, was started in 2004. Although KHR-3 was based on the design of KHR-2. Many parts of the frame were improved: leg, waist, upper body. HUBO is 56kg heavy and 125cm tall, has 10 fingers, 2 eyes (vision camera), and 41 DOF. It contains a battery pack for power source and works for about fifty minutes after charging. It is also possible to operate by remote-control wherever a network is connected. Due to the simple structure, KHR-3 came to have mechanical stiffness and minimized mechanism uncertainty [Park 2007].

Of course, many other humanoid robots are being created in different researching centres around the world, and many more bipedal walking robots. Despite the grandeur and publicity surrounding humanoid robots today, they are still a very primitive form of the envisioned artificial humans.

The main comparative characteristics of the humanoid robot, as mentioned above, are weight, height, number of degrees of freedom and the computational power. There is an evidential relationship between the size (height) of the humanoid robot and its weight. In the case of humanoid robots of human-like size, the best relationship between weight/height/DOF is displayed in ASIMO, HRP-2, WABIAN 2 and KHR-3.However, it is rather difficult to compare their control and information systems because of a lack of information available in this field. It

can be assumed that modern robots have the contemporary control system based on available up-to-date hardware and software.

The following tables show the comparative characteristics of the most significant full size humanoid robots developed up till the present day.

Name	WABIAN	WABIAN R IV	WABIAN 2
Research Centre	University of Waseda	University of Waseda	University of Waseda
Country	Japan	Japan	Japan
Year	1995	1997	2005
Weight (kg)	107	131,4	67.5
Height (mm)	1662	1890	1475
Actuators	Servomotors DC	Servomotors DC	DC Servo Motors Harmonic Drive Gear Timing-belt/Pulley
Control Unit	Walk / Operating Control Unit, Pentium PC/AT CPU board	Walk / Operating Control Unit, Pentium PC/AT CPU board	Walk / Operating Control Unit, Wireless Transmission PCI CPU board, QNX OS
Power Supply	Cable	Cable	Batteries
DOF: Head (Eyes + Neck)	2 x 2 + 2	2 x 2 + 4	3
DOF: Arm	7 x 2	7 x 2	7 x 2
DOF: Hand	3 x 2	3 x 2	3 x 2
DOF: Trunk	3	3	2 + 2 in Waist
DOF: Leg	2 x 2	4 x 2	4 x 2
DOF: Foot	1 x 2	2 x 2	3 x 2
DOF Total	35	43	41

Table 2.2: Comparative characteristics of the most significant humanoid robots: WABIAN, WABIAN RIV, WABIAN 2

Name	P2	P3	ASIMO (New ASIMO)
Research Centre	HONDA	HONDA	HONDA
Country	Japan	Japan	Japan
Year	1996	1997	2000 (2004)
Weight (kg)	210	130	52 (54)
Height (mm)	1820	1600	1200 (1300)
Actuators	DC Servo motors, Harmonic Drives	DC Servo motors DC, Harmonic decelerator	Servo Motors, Harmonic Speed Reducer + Drive unit

Control Unit	Walk / Operating Control Unit, Wireless Transmission	Walk / Operating Control Unit, Wireless Transmission	Walk / Operating Control Unit, Wireless Transmission, Pentium-III M 1.2 GHz
Power Supply	Batteries	Batteries	Batteries
DOF: Head (Eyes + Neck)	0	0	2
DOF: Arm	7 x 2	7 x 2	5 x 2
DOF: Hand	2 x 2	1x 2	1 x 2
DOF: Trunk	0	0	0
DOF: Leg	4 x 2	4 x 2	4 x 2
DOF: Foot	2 x 2	2 x 2	2 x 2
DOF Total	30	28	26 (34)

Table 2.3: Comparative characteristics of the most significant humanoid robots: P2, P3, ASIMO

Name	**H6**	**H7**	**JOHNNIE**
Research Centre	University of Tokyo	University of Tokyo	Technical University of Munich
Country	Japan	Japan	Germany
Year	2000	2001	1998
Weight (kg)	55	58	40
Height (mm)	1370	1470	1800
Actuators	DC Servo motors, Harmonic Drives	DC Servo motors, Harmonic Drives	DC Servo motors, Light reduction gearboxes
Control Unit	PentiumIII-750MHz, RT-Linux, wireless Ethernet	Dual Pentium III 1.1GHz, RT-Linux, wireless Ethernet	External PC, CAN bus
Power Supply	Batteries	Batteries	Cable
DOF: Head (Eyes + Neck)	5	2	0
DOF: Arm	7 x 2	6 x 2	2 x 2
DOF: Hand	1 x 2	1 x 2	0
DOF: Trunk	0	0	1
DOF: Leg	4 x 2	4 x 2	4 x 2
DOF: Foot	3 x 2	3 x 2	2 x 2
DOF Total	35	30	17

Table 2.4: Comparative characteristics of the most significant humanoid robots: H6, H7, JOHNNIE

Name	HRP-2P	KHR-2	HUBO (KHR-3)
Research Centre	KAWADA Industries	KAIST	KAIST
Country	Japan	Korea	Korea
Year	2002	2003	2006
Weight (kg)	58	56	56
Height (mm)	1540	1200	1250
Actuators	Motors DC	Motors DC	Motors DC
Control Unit	2 boards CPU Pentium III 1GHz ART-Linux OS	Real Time distributed control using CAN bus. Windows 2000 with RTX	Real Time distributed control using CAN bus. Window XP - RTX
Power Supply	Batteries	Batteries	Batteries
DOF: Head (Eyes + Neck)	2 x 2	2 x 2	2 x 2
DOF: Arm	5 x 2	6 x 2	6 x 2
DOF: Hand	2 x 2	5 x 2	5 x 2
DOF: Trunk	0	1	1
DOF: Leg	4 x 2	4 x 2	4 x 2
DOF: Foot	2 x 2	2 x 2	2 x 2
DOF Total	30	41	41

Table 2.5: Comparative characteristics of the most significant humanoid robots: HRP-2P, KHR-2, HUBO

By comparing presented humanoid robots' configurations it is possible to make some conclusions about the tendencies in this area of development. The basic parameters of humanoid robots such as height (Fig. 2.27(a)), weight (Fig. 2.27(b)) and number of D.O.F. (Fig. 2.27(c)) change over time. Design concepts tend towards reducing the weight and height and increasing the number of degrees of freedom of the robot. By analysing actual humanoid robots, it is possible to conclude that from a design perspective, the best humanoid robots are 1300 mm in height and 60 kg in weight and have at least 35 D.O.F. With these measurements , the ideal relationship between the mechanical stiffness and movement accuracy can be achieved.

Moreover, Honda's criterion for the ideal humanoid robot size was based on the idea that it could cooperate comfortably with humans. The robot's size should be designed in order to allow it to operate freely in the human living space and to make it people-friendly. This size allows the robot to operate light switches and door knobs, and work at tables and work benches. Its eyes are located at the level of an adult's eyes when the adult is sitting on a chair. A height of 120cm makes communication easier.

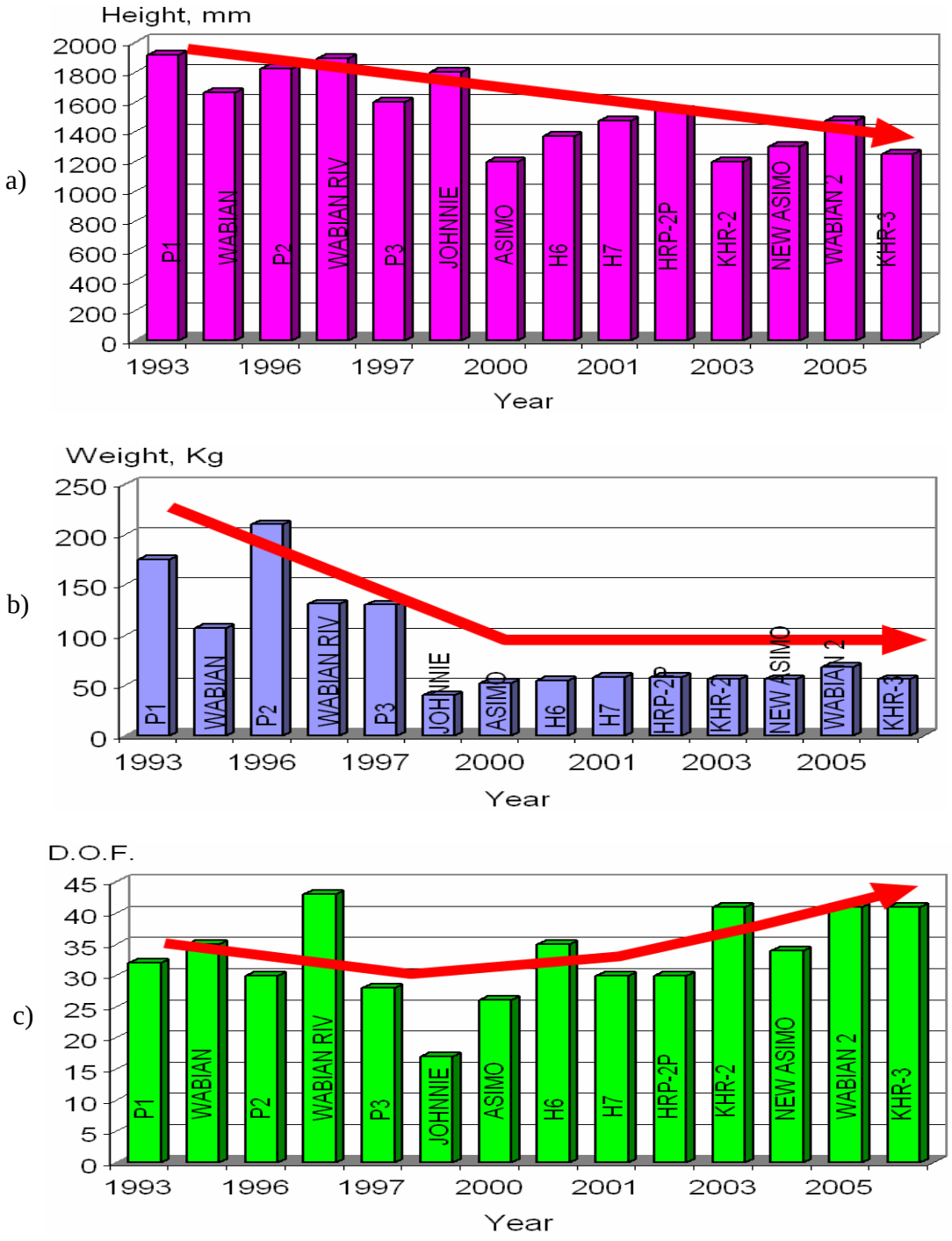


Fig. 2.27: Evolution of basic parameters of humanoid robots a) Height, b) Weight, c) D.O.F.

Once the main characteristics of the humanoid robot are determined, advantages and disadvantages of the proposed robots can be considered. Firstly WABIAN series as well as the P2 and P3 prototypes of Honda have a lot of DOF that achieves rather good results in bipedal movements. However due to heaviness there are restrictions in its use in human-robot interaction

within the human environment. Furthermore, these robots just as JOHNNIE humanoid are not totally autonomous because of their power supply by cable. H6 and H7 humanoids can walk stably, but there is not a lot of information about other important abilities, such as running or human interaction. WABIAN 2 humanoid robots don't have any of the indicated disadvantages of its predecessors and can be considered as one of the best contemporary robots. ASIMO has a lot of advantages from a technological point of view, but a major part of the information regarding this model is confidential due to the politics of the Honda Company and therefore cannot contribute to the development of future humanoid projects. The KHR-3 robot is very well designed and different innovations in hardware and software design were implemented in its production, but it can not be the reference point for humanoid robotics yet, because of the rather short time it has spent in the "upper league" in the humanoids field. The HRP-2P robot is autonomous and can walk very stably.

The main advantage of HRP-2P is the number of prototypes constructed and collocated in the best research centres in the world. This demonstrates the reliability of its mechanical and electrical design and the possibility to develop user's own tasks with the humanoid because the control software is of open architecture (OpenHRP). It makes the HRP-2 humanoid one the main platforms for further popularization and development of more complex interactions with humanoid robots. Part of the research conducted to inform the writing of this Ph. D. thesis was experimentally tested and implemented with this experimental platform.

2.5 Current humanoid research

Humanoid research incorporates an extensive range of areas, across many disciplines and fields of study. Each of these disciplines, in their own way, contributes towards the development of the artificial human.

From our current experience, even to build a basic humanoid (for example the only movement being walking) is a complex and expensive task. As mentioned above, it took 10 years of research by Honda Motors to develop P2 and more to produce ASIMO. Also they have almost certainly invested billions of US dollars in these projects. The journey is not finished yet, only a small step has been taken along the road to the creation of a real artificial human.

Due to the complexity of the humanoid system, the research to date is still rather fragmented. There are several distinct research areas in the development of the humanoid robot. The major research areas today are bipedal locomotion and artificial intelligence. Both of these topics are related to the control of a humanoid robot and will be considered in more detail in the following sections.

Since humanoid research involves a multitude of disciplines, many valuable engineering and scientific results could be obtained through it. There are many benefits of humanoid research, some direct outcomes and others, indirect. Some of these could be realized in the relatively short-term (less than 10 years) while others may take a much longer time. In this chapter, the benefits of the research are categorized into three areas: technological, scientific, and economic [Xie 2003].

2.5.1. Technological impact

Humanoid research can contribute to the growth of technologies and result in useful systems that are able to perform certain tasks. As demonstrated in previous sections, the initial aim of humanoid research was to build better orthosis and prosthesis for human beings. For example, we can cite powered leg prosthesis for the neuromuscularly impaired, ankle-foot orthosis, biological realistic leg prosthesis and forearm prosthesis. Although the transfer of knowledge from biped walking research to prosthesis development is not very clear and direct, there is a rapidly growing trend towards closer cooperation. Collaboration between humanoid researchers and prosthesis developers will be required to make major steps forward in this very important area.

The humanoid robot can be a test platform for many state of-the-art technologies, such as vision, tactile sensors, actuators, materials, machine learning algorithms, etc. New technologies will always involve new control architectures, actuator systems, AI strategies, and so on. Humanoid robots can also serve as a platform for technological education. Due to its multidisciplinary nature, there are many possible uses for humanoids in this field.

Today's engineering education emphasizes the integrative aspects of various engineering disciplines, or Mechatronics. Mechatronics originated in Japan in the 1970s and is gaining worldwide attention because of the importance of integrating different, but inter-related engineering disciplines. A formal definition of Mechatronics can be stated as follows [Bradley 91]: "Mechatronics is the study of the synergetic integration of physical systems with information technology and complex decision-making in the design, analysis and control of smart products and processes".

The humanoid robot design project, being a combination of mechanical, electronic, control, information, perceptive and cognitive systems, is an ideal platform for implementing the essence of Mechatronics and educating students in control algorithms, mechanical design, motion control, machine vision, system integration, etc. Technological education can also be organized in the form of games and competitions. For example, RobotCup and FIRA Robot Soccer competitions are two international robotics game organizations which produce annual humanoid-robot, soccer-playing competitions with the participation of a multitude of robotics laboratories from all over the world.

2.5.2. Scientific (research) impact

Here, we will refer to those scientific discoveries which may not have an immediate direct application or high commercial value, but are of interest to scientists performing fundamental research. Monetary return is not the focus for such research, although the discoveries may have commercial value in the future.

There are numerous benefits resulting from this type of humanoid robot research. The humanoid robot can be an excellent test platform for various hypotheses or models. For instance, researchers need to understand the human body structure, behaviour (biomechanics) and walking gait to build and study humanoid robots. On the other hand, our attempt to model and replicate the human body leads to a better understanding of it. The knowledge gained in such studies can be applied to improve rehabilitation procedure for patients who have lost their walking or

manipulative abilities. The outcomes to these applications could also be used to further improve humanoid systems.

Besides biped locomotion, which of course drives research into humanoid robots, an additional force which stimulates research in this fascinating field is the drive to develop an intelligent (physical) system.

Intelligence and the ability to produce intended results are unique attributes associated with humans. One possible definition of intelligence is as follows [Xie 03]: “Intelligence is the ability to link perception to actions for the purpose of achieving an intended outcome. Intelligence is a measurable attribute, and is inversely proportional to the effort spent in achieving the intended goal”. The goal of artificial intelligence (AI) [Sage 90] “is the development of paradigms or algorithms that require machines to perform cognitive tasks, at which humans are currently better”. The main task is to make robots able to perceive, make their own decisions, learn and interact with human beings.

An artificial body requires artificial intelligence in order to adapt to changes in the environment to obtain better performance of the assigned tasks. Undoubtedly, the humanoid robot is a perfect research platform for the study of the embodiment of artificial intelligence within an artificial body. It is an ideal testing ground for computational theories or models derived from the study of the human brain’s mechanisms.

2.5.3 Economical impact

This subsection discusses the economic or commercial value of humanoid robot research. Up until today, the greatest consumer product created has undoubtedly been the auto mobile. It is technologically complex and is not cheap, but it provides us with unprecedented mobility, increases our sense of social status, and offers us high-quality entertainment experiences among other things. Without the auto mobile, there would be no efficient transportation in our modern world. Now, we are facing the question: Could a humanoid robot become the next great consumer product? Looking at the evolution of the computer as a consumer product, the following trends may be identified:

- The microprocessor is becoming smaller and consuming less electrical power, while computational power is increasing.
- The computer has more functions and better performance while the price is constantly decreasing.
- The computer is constantly improving its cognitive abilities: soft computing, speech recognition, auditory communication, visual communication, visual perception etc.

The evolution of the computer coincides with the evolution of the humanoid robot, which is characterized, by the analogy with the computer industry, by the following clear trends:

- The robot starts to have legs to perform human-like, biped walking.
- The robot’s arm and hand is becoming more compact, and it has increased sensing ability.
- The robot has a head, which may influence the design of future computer monitors, and provide the means for facial expressions and perceptive sensory input and output.

- The robot is becoming increasingly intelligent. The incorporation of demanding computational power and cognitive abilities enables the robot to perform real-time interaction with its environment and humans.

The parallel development of the computer and robot industries will contribute to the emergence of the humanoid robot as the next great consumer product.

It is important to mention that recently various commercial opportunities have arisen from humanoid robot research. An immediate one is in the increasingly popular toy sector. Nowadays computerized games are very popular. These games basically are pre-programmed interactions between users and artificial creatures in a virtual world. With the humanoid robot, or an animal-like robot, computerized games will take on a whole new dimension: a pre-programmed interaction with artificial creatures in the real world. Although it brings new opportunities, complex problems also arise in relation to interaction of a robot with human beings. Taking into account the success of Sony's AIBO robot dog, there is good potential for a humanoid toy robot too. Several Disney attractions employ the use of animated artificial bodies. These robots that look, move, and speak much like human beings, are now in some of their theme park shows. These artificial bodies look and move so realistically that it can be hard to recognize from a distance whether they are actually human or not. Although they have a realistic look, they have no intelligence or any kind of walking autonomy.

In the long term, humanoids can be developed as domestic helpers. It would be able to carry out normal household work, act as a security guard, and look after the elderly or children. As one of the effective ways to lower the cost of services is to increase the accessibility of these services, another possible implementation of a humanoid robot could be robot-assisted services (healthcare, personal services) in the home. This would be possible, if humanoid robots were more technologically advanced. By simply activating the appropriate programs, a humanoid robot could instantaneously be configured as an “expert” in the necessary service field.

A humanoid robot could also be used in robot-assisted education at home. In the future, an alternative to private tuition may be the use of humanoid robot-tutors, with pre-programmed and selectable knowledge and skills. One obvious advantage to using the humanoid robot as a tutor is the ability for the same humanoid robot to be configured as a tutor in different disciplines, as well as at different skill and knowledge levels.

Everything stated above is a potential multi-billion business that can easily overtake the automotive and aerospace industries in the future. Companies - world technological leaders supported by the government of their country have already begun the competition and conquest of a new emergent market.

2.6 Ethical concerns of humanoid research

The development of humanoid intelligence and capability raises some serious ethical questions. Most of them apply not only to humanoids, but to the robotic field in general.

Some people think that humanoids can continue to learn and evolve to a point where they will break away from human command and possibly revolt, or that their “upbringing” can determine their “personality” (e.g. a selfish, tyrannical person will produce a similar android). Nowadays when we are facing the possibility to coexist with intelligent machines, Asimov’s laws of

robotics are not enough to set ethical guidelines concerning the roles and functions of robots in the modern society. Key problems include ensuring human control over robots, protecting data acquired by robots and preventing its illegal use. For example, to whom should be awarded the patents of an invention done by a robot? Or, who is responsible when an intelligent machine fails, commits a crime, or does something it shouldn't do?

In the future, when humanoid robots have the ability to reason and have feelings, other types of questions will be raised. What would be the difference between human beings and humanoid robots' rights? Could a person destroy or make robots his/her slaves? Should a humanoid robot, to protect his existence, be permitted to cause damage to humans or other robots? Why would we ever want to design a robot with the ability to have freewill or even a sense of self preservation?

Nowadays, other concerns are emerging mainly with the introduction of humanoid robots in tasks that till now were done only by human beings, for example, jobs such as a security guard, a policeman or a soldier. Humanoid robots for factories are also being developed. This may cause people lose their jobs to robots that work with lower costs and higher productivity.

As we make robots more intelligent and autonomous, we need to consider how to govern their behaviour and how much freedom to grant them - so-called roboethics. Roboethics is the ethics applied to Robotics, guiding the design, construction and use of the robots.

Robotics is rapidly becoming one of the leading fields of science and technology, so that very soon humanity is going to coexist with a totally new class of technological artefacts: robots. It will be an event rich in ethical, social and economic problems. The name Roboethics (coined in 2002 by G. Veruggio) was officially proposed during the First International Symposium of Roboethics (Sanremo, Jan/Feb. 2004), and rapidly showed its potential.

"Roboethics is an applied ethics whose objective is to develop scientific, cultural, technical tools that can be shared by different social groups and believes. These tools aim to promote and encourage the development of Robotics for the advancement of human society and individuals, and to help preventing its misuse against humankind."

Now, for the first time, we find ourselves facing the challenge of replicating our own intelligent and autonomous entity. This obliges our scientific community to examine closely the concept of intelligence from a cybernetic point of view. In reality, complex concepts like autonomy, learning, evaluation, decision making, freedom, emotions, and many others must be analysed, taking into account that any given concept need not have the same meaning for different species such as humans, animals, or machines.

Therefore, it would seem not only necessary but also natural that robotics should affect many other disciplines, such as Logic, Linguistics, Neuroscience, Psychology, Biology, Physiology, Philosophy, Literature, Natural History, Anthropology, Art, and Design. It could be said that Robotics unifies the so called two types of knowledge – the sciences and humanities. The effort to design Roboethics makes this distinction redundant.

All f this means a set of guidelines on the use and development of robots, especially human-like humanoids is more necessary than ever. Different international bodies have started to consider these ethical concerns provoked by modern developments in robotics. The European Robotics Research Network is also drawing up a set of rules regarding robotics. This ethical roadmap has been assembled by researchers who believe that robotics will soon come under the same scrutiny as disciplines such as nuclear physics and Bioengineering. A draft of the proposals

said: “In the 21st Century humanity will coexist with the first alien intelligence we have ever come into contact with - robots. It will be an event rich in ethical, social and economic problems.” [Veruggio 2007]

2.7 Conclusion

Nowadays, robots are widely used in industry. In everyday life, we might make use of service and already even personal robots sometimes. Therefore, the research on humanoid robots that can combine all of these functions is one of the hottest topics in engineering society today. Because of its anthropomorphic structure, a humanoid robot can work directly in the same environment as a human. Moreover, humanoids can use the same tools and operate same machines as humans, but more efficiently. Also they can be used in conditions unsuitable for human such as in space or radioactive zones.

Nevertheless, the development of humanoid robots still remains a very complex engineering task that requires new approaches in mechanical design, electronics, software engineering and control. During its evolution, the humanoid robot has been transformed from a ponderous bipedal mechanism into a human-like artificial body with rather good walking capacities.

Nowadays, humanoids have a great impact on many areas of engineering, across many disciplines and fields of study in robotics. Also, the rapid development of humanoid intelligence and capability raises some serious ethical questions which are a topic of concern and interest for many philosophical and technical researches.

We are still in the beginning of the humanoid robotics era. There are many problems in the area of humanoids to be solved and one of the principal ones is the control framework for stable bipedal locomotion. The following chapters of this Ph. D. thesis will provide a possible solution to this very problem - the design of control architecture for bipedal locomotion of humanoid robots.

Chapter 3

Concepts of the Robot’s Control Architecture

CHAPTER 3

Concepts of the Robot's Control Architecture

3.1 Introduction

Before embarking upon the design of an open control architecture for a humanoid robot, which is a main topic of this Ph. D. thesis, let us make a short study and classification of the basic concepts of a control architecture design, principally for contemporary robotics systems. Firstly, this chapter will address the question of what control architecture is, and also consider how robot control architecture design should be realized. After these questions are clarified, some case-studies of the most representative control architectures for different types of contemporary robots will be examined.

The traditional approach to control architecture design for robots requires us to implement different processing modules, realizing a variety of functions. These functions can be divided into basic and high level functions of intelligent control. Basic control functions include the servo control of all the articulations and the generation of secure motion trajectories, including angular and Cartesian interpolation. In the case of teleoperation, control is limited and only manages communication with the master device to follow the motion commands of the human operator. Intelligent or high level control functions can allow the realization of partially autonomous operations such as task planning, navigation and the creation of precise, representation of environment, which can be used for auto localization. Moreover they provide a high level of perception by fusing the data provided by different sensors and allowing implementation of the advanced capacity to react to non predicted conditions of the environment, or the presence of obstacles. Finally, the high level control provides for interaction between the user and the robot through the Human Machine Interface (HMI). These basic and advanced (intelligent) functions are not isolated from each other but are integrated as different modules in an overall scheme determining the control architecture of the robot. When incorporating the intelligent control functions into the control architecture it is usually necessary to study some additional aspects such as the efficiency of task execution, the capacity to execute multiple tasks, the capacity to react to different events (internal or external), or emergency. Even in teleoperated control schemes it can be useful to incorporate some techniques for adequate autonomous reaction in dangerous situations and in order to avoid collisions.

Modern tendencies in robotics attempt to provide robots with the maximum autonomy possible. The main objective of modern research is to achieve a level of autonomy where no external human intervention for decision making is needed. In this case all modules of

perception, data processing and decision making should be implemented on-board the mobile robotics platform. Also needed are the control hardware and software architectures which also physically take their place on-board the robot. This creates different problems related to the physical distribution of all the components inside the robot's body, which should also be taken into account during the design process. The autonomy of the robot generates other problems related to localization and navigation. During last decade different control schemes providing solution to these problems have been proposed.

We can understand the control architecture to be the way of organizing and providing the relations between different components of the system (hardware and software). In the following section a more precise definition of the control architecture will be proposed. Generally speaking, the architecture groups all on-board and external components of the robot inside the hierarchical structure in different layers, determining the way of providing different functionalities.

The main objective of this chapter is to introduce universal concepts for the design of the motion control architecture for humanoid robots. First of all, the general concepts for system architecture design will be presented and the basic existing types of control architectures and their different modes of operation will be studied. The principal characteristics of the control architecture should comply with the functional and flexibility requirements. After these requirements are established, the work will be focused on the specific type of control framework – the motion control. The classification of different types of robots according to its motion abilities was proposed in the previous chapter. Based on this classification table, each type of robot will have its own particular motion control features. Through the analysis of particular parts of the motion control frameworks for different robotics systems, we can draw conclusions and establish rules governing the open motion control architecture design for humanoid robots, which will be described in detail below.

3.2 System Architecture Definition

As mentioned in the previous section, system architecture is the design, or set of relations, between the parts of a system. There is no strict definition of which aspects constitute a system architecture, and various organizations define it in different ways, including:

"The fundamental organization of a system, embodied in its components, their relationships to each other and the environment, and the principles governing its design and evolution". From ANSI/IEEE 1471-2000.

"The composite of the design architectures for products and their life cycle processes". From IEEE 1220-1998 as found in their glossary.

"A representation of a system in which there is a mapping of functionality onto hardware and software components, a mapping of the software architecture onto the hardware architecture, and human interaction with these components". From the Carnegie Mellon University's Software Engineering Institute as found in its glossary.

The topic of this thesis is the study and design of humanoid robot architecture. A robot in general, is usually a mechanical or virtual, artificial creature. A robot is an electromechanical system, which needs to be managed by special software and hardware control agents. This Ph. D.

thesis is focused mainly on control problems, thus, hereinafter the term "systems architecture" will be understood to mean the architecture employed to control a robot.

As is clearly seen from the above definition, a system control architecture has two basic parts, hardware and software infrastructures, which are closely related to each other. A control system generally refers to a programmable hardware machine and its included program. A systems architecture is defined as one concerned with the complete device, both hardware and software and, more particularly, all of the interfaces of the device, including that between hardware and software, and especially between the complete device and its user. The hardware architecture deals (more or less) exclusively with the hardware device (CPU, controllers, sensors). The software architecture deals (more or less) exclusively with the software program (containing the control laws and algorithms). The systems architecture is responsible for seeing that the software program is capable of properly running within the hardware device, and that the system composed of these two entities is capable of properly performing its intended function and interacting with its external environment, especially with the user.

Robot control architecture makes use of elements of both software and hardware and is used to enable the design of such a composite system. Control architecture may be viewed as a "partitioning algorithm", which partitions all of the system's present and possible future requirements into a workable set of clearly bounded rules, determining subsystems of architecture. This partitioning scheme should be exclusive, inclusive, and exhaustive. A major purpose of the partitioning is to arrange the elements in the subsystems so that there is minimum communication needed between them. A good architecture provides an easy mapping for the user's requirements. Ideally, a mapping also exists from even the smallest element to every requirement and test.

The hardware control architecture is an abstract representation of an electronic and/or an electromechanical device which is capable of running a fixed or changeable program [Assif 98], [Zimmerman 2008]. It is generally some form of analogue, digital, or hybrid electronic computer and/or its electronic and mechanical accessories.

The software architecture for control a computing system is the structure or structures of that system, which comprises software components, the externally visible properties of those components, and the relationships between them [Bass 2003].

It should be noted that robust control architecture exhibits an optimal degree of fault-tolerance, backward compatibility, forward compatibility, extensibility, reliability, maintainability, availability, serviceability and usability as necessary and/or desired.

3.3 Control Architecture Requirements

When designing the control architecture for a robotics system, first of all it is necessary to specify some general requirements for all types of robots. The basic aspects that influence the control architecture are as follows (note different robotic systems may need further requirements to be developed): efficiency, programmability, adaptability, portability, the level of autonomy and evolution capacity. Let us consider these requirements in more detail.

Level of autonomy

The autonomy of a robot can be considered as its capacity to make its own decisions based on the information that it receives from the sensorial system about the environment and task. As it was mentioned in previous sections, modern tendencies in robotics systems provide the replacing of direct and teleoperated control by autonomous behaviours. Also, some kind of dual control mode can be implemented combining both of these functions. Dual control approach has an impact on the design process of the control architecture, especially on the upper control level. The main problem that needs to be solved is how to ensure an adequate distribution of autonomous actions and human operated actions. In this context, the most important criterion seems to be the user-friendliness. Figure 3.1 shows the task distribution between user and robot for different levels of autonomy of the system.

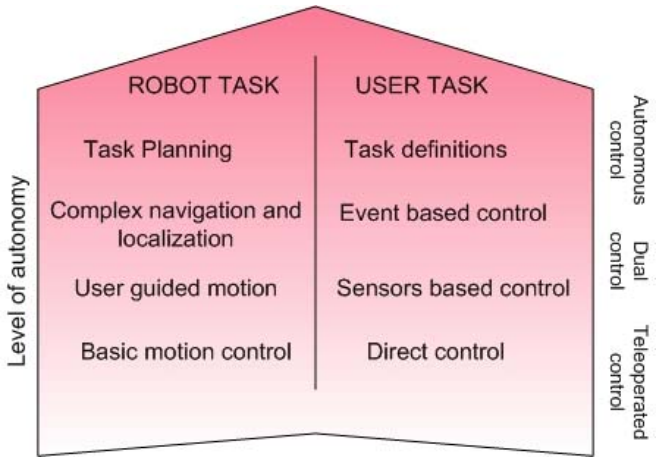


Fig. 3.1: Task distribution between the user and the robot [Jardón 2006]

It should be noted, that the high level of autonomy of a robot is achieved by intense activity on all control levels, especially on the upper one. This maximises the effectiveness of user ability and input, thus decreasing his or her effort. The user is only required to specify upper level objectives and tasks. Meanwhile, at the bottom level of autonomy, the main control effort is supported by the user while the robot performs lower level control (joint control) tasks. Interaction is maintained by teleoperation and requires greater user involvement in the control process. Dual controls can be utilized, bridging the two extremes of user control and robot control. . Usually these controls are called a semi-automatic control schemes. Essentially, the level of autonomy depends on the concrete task, therefore the control architecture should provide for adjustment according to the needs of the user and application.

Contemporary robots can be used in a variety of tasks, manifesting dual control abilities. For example, in order for a robot to pick up some objects in a domestic environment (such tasks as books from a bookshelf) the user can control the manipulator of a robot directly or can specify adequately the goal while the robot plans the motion. Other applications, for example bipedal locomotion of a humanoid robot, should be executed in a full automatic mode. The user specifies the position, and the robot computes the trajectory to generate human-like bipedal locomotion. On the other hand, there does exist a semi-automatic control mode. If the environment is non-structured and presents obstacles, the operator can control its avoidance of these, correcting on-line the goal position, while the bipedal locomotion generation procedure remains automatic.

Concluding, it can be stated that increasing the autonomy of a robot simply means reducing the need for human supervision and intervention. Autonomy is interpreted relative to the amount of on-line (while the robot is operating) involvement of human operators.

Efficiency

The efficiency denotes both realization of a task (the time needed for achieving a goal and precision of operations) and resources implemented in order to achieve this task. Here by resources we mean the energy, computational power, hardware and software load. For example, sometimes it is not efficient to implement autonomous control (usually more resource-required) in a case when semi-autonomous operation can give the same results and meets the demands of the user and task.

Efficiency can be achieved by using adequate models of a robot and environment and implementing optimal algorithms of motion planning using these models. Also, it is necessary to define some limits for the time and resources needed to realize several specialized control functions.

Programmability

Programmability is the capability within hardware and software to change, to accept new sets of instructions that alter its behaviour. Programmability generally refers to architecture logic, but it also refers to design of the user interface which includes the choice of menus, buttons and dialogues. When designing the control architecture of a robotics system it is necessary to decide how it will be programmed. Programmability defines the execution of a task, the possibility of multi task functioning of a system, the way of planning actions and represents the state of a robot and the environment.

There are different levels of human operator intervention in a robot's functioning in order to program a task. On the one hand, all operations can be completely sequenced and all parameters can be specified by the user. This method requires strong user knowledge of the system and programming environment and is widely used in industrial and research applications, but is not suitable in major areas of assistance and personal applications. These applications require architecture which allows the user to specify only the main goals, while all planning and motions are executed automatically.

Adaptability

This requirement is related to the flexibility, or the capability of the robot to work in different, unknown environments, and adapt to unexpected changes. Adaptability can be considered as the modification of behaviour in compliance with the given environment. In another words, it is the robot's capacity to analyse situations and produce (or select) appropriate actions for any changes in the environment and general working conditions using sensorial information and models. Basically, autonomous and semi-autonomous architectures comply with this requirement.

Reliability

In general, reliability is the capacity of a system to perform and maintain its functions in routine circumstances, as well as hostile or unexpected circumstances. The IEEE defines it as "the ability of a system or component to perform its required functions under stated conditions for a specified period of time." In the case of robotics, it can be related to the absence of system faults in hardware and software components.

In some critical fault systems, when error can provoke disastrous effects, it is possible to implement some redundancy. Software redundancy can be easily achieved by implementing different modules with the same functionality, working simultaneously and realizing the task. In the case of a critical error or a fault in one module, another one can complete the operation or at least avoid possible accidents. Hardware redundancy can be implemented, duplicating critical elements. In the case of a fault, alternate hardware will assume the function of a disabled module. Moreover, the duplication of sensors helps to decrease the uncertainty in perception processes.

Some mobile systems have very restricted availability of space and don't allow for the insertion of additional hardware systems. Moreover, the increase in computational power (in the case of software duplication) and spatial load (in the case of hardware duplication), stands in contradiction with the efficiency requirements discussed above. More efficient in this case, seems to be a generalized supervising system, allowing for a decrease in the consequences of the non-regular functioning of a system.

Evolution capacity

This requirement is important because contemporary research projects continue over several years, and continuous progress in electronics and software allows for revision and updating of technologies, equipment and components. This requirement has elements in common with the next point to take into account – portability of control architecture.

Portability

Firstly, this term was applied in computer science. Porting is the process of adapting software so that an executable program can be created for a computing environment that is different from the one for which it was originally designed (e.g. different CPU, operating system, or third party library). Nowadays, this term can also be used in a more general way to refer to the changing of software/hardware architecture to make them usable in different robotics platforms.

The architecture is portable when the cost of porting it to a new platform is less than the cost of implementing it from scratch. The lower the cost of porting architecture, relative to its implementation cost, the more portable it is said to be.

However, currently, very fast evolution of hardware and software leads to rather limited portability of robotics platforms. Basic components become obsolete very quickly and the replacement of hardware and software components for modern and more powerful ones leads to the change of a control strategy and sometimes the whole control architecture of a robot.

To conclude, let us take the notion that some requirements presented below sometimes clash with each other. For example, efficiency (planned motions) and adaptability (reactive motions) of a control architecture which will be discussed in a following sections. In this case, the architecture should be designed by evaluating efficiency of execution of a determined task using determined resources. Also, it is necessary to evaluate its capacity of multi task execution, evolution capacity and reliability.

3.4 Design Levels of Control Architecture

Before starting development of any type of control architecture it is necessary to define basic levels of a designing process. Usually, the uppermost design level is committed to the *conceptual design* of the system. It results in an overall architecture and fundamental functions of the designed system. Also, decisions of which functions will be implemented in the hardware, either analogue or digital, and which of them will be implemented in the software. The behavioural description at this level corresponds directly to the introductory specification of the system. The topological description, usually in the form of a block diagram, indicates at this level the fundamental `causes-and-effects' within the system.

At the *functional design level*, the system architecture and functions are refined into more detail. The modules at this level are assumed to interact in the form of physically dimensionless continuous-time, continuous-level signals, or signals discretized in time, level, or both. The module interactions are governed by algebraic rules. The topological description used at this design level is mostly in the form of block diagrams with blocks characterized by equations, transfer functions, tables, etc. Typically, the design of control strategy and control architecture, either analogue or digital, is a matter of functional-level design.

The *physical design level* is concerned with hardware and software implementation of the system architecture and functions. Design modules take the form of real system components or physical effects and their mutual interactions.

At the *technological design level* the assembly or another manufacturing process for the system and for its components is designed.

3.4.1 Conceptual design. Basic Modes of operation in robotics systems

In this section, in order to provide the conceptual design of a system, let us consider the questions related to the execution of the determined task using a robot and its affect on the design requirements for the control system of a robot. As it was described in the previous section, modern tendencies in robotics provide the rise in the level of autonomy of a robot. It leads to the question of how we can establish the relation between the autonomous system and the human master. Here we have three different ways to design a control system relating to the task that a robot should execute and the level of its autonomy [Vertut 85]. In the figure 3.2 the basic direct remote control is presented. The human master or the user generates direct movement orders needed to achieve a task and the robot only executes the motion and provides sensorial data acquisition of the environment. The user should process this information and close the control loop by taking control decisions.

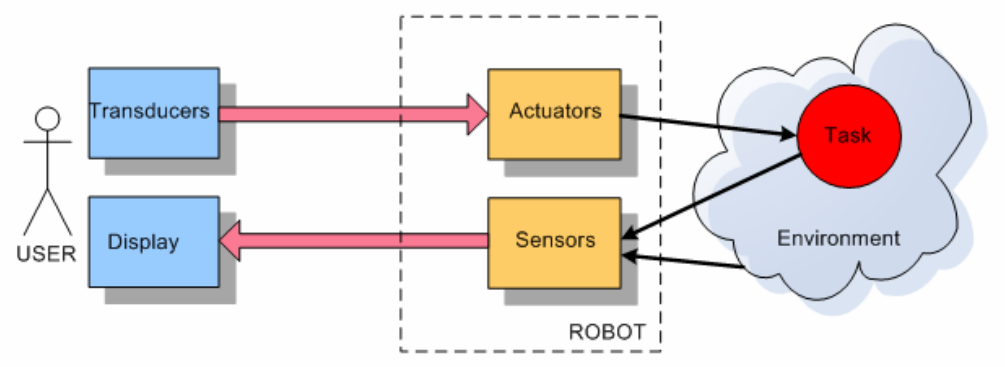


Fig. 3.2: Direct control scheme

Figure 3.3 shows the control teleoperated or remote teleoperation. The user can control the robot from a very long range of distances due to the use of a communication bus which transmits the information between the master and the robot. The difference with the previous case is that because of the presence of different types of sensors and a communication bus, the user has information about the environment and the state of the robot without being located physically near the working scene. There are two additional control loops. The local control loop which belongs to the robot executes the bottom control services and functions. The remote control loop in the operator's side provides the mapping of the environment, the state of the robot and the task for the user. These two additional loops will characterize whole control system behaviour.

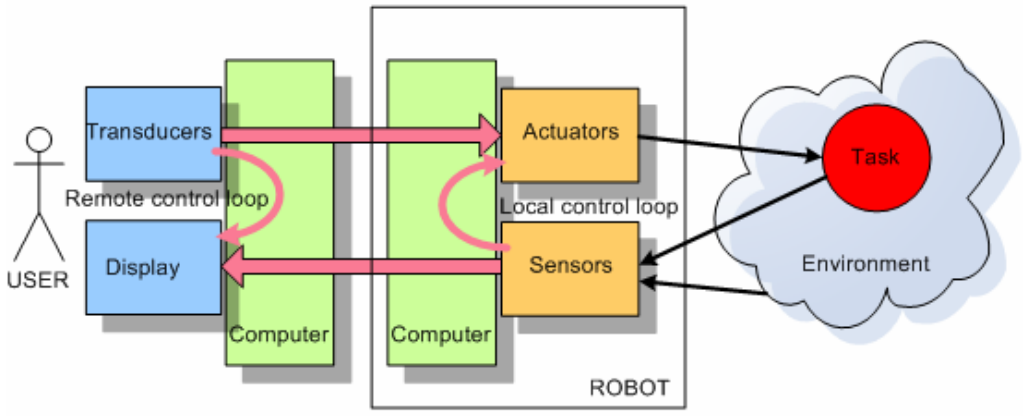


Fig. 3.3: Control scheme with teleoperation

The classical control scheme with teleoperation needs the continuous attention of the remote user in order to manipulate the robot. In order to facilitate the task for the operator, the robot is provided with some autonomous functionality. The local control loop increases its significance and the volume of information needed for the user to control the process decreases. Now user commands are not bottom level commands like positions or force references but become upper level commands raising the level of abstraction of the control process. The control procedure at the local control loop assumes the processing of sensorial information in order to execute autonomous tasks.

The extreme case of the autonomous mode of operation of a robotics system is presented in figure 3.4. The robot executes the goal in a completely autonomous way and the user only observes the result of its operations. The operator supervises the execution of the previously planned tasks providing the information about essential objectives and restrictions. Moreover, as the working environment usually is non-structured and changes dynamically, there can be some unexpected situations which were not programmed in the plan of actions of the robot. Thus, the robot is not able to solve these situations correctly without the user's intervention. Furthermore, it is very expensive from the computational point of view to provide a robot with totally autonomous behaviours and on the whole is physically impossible to program the appropriate behaviour of the robot in every possible situation of its operation.

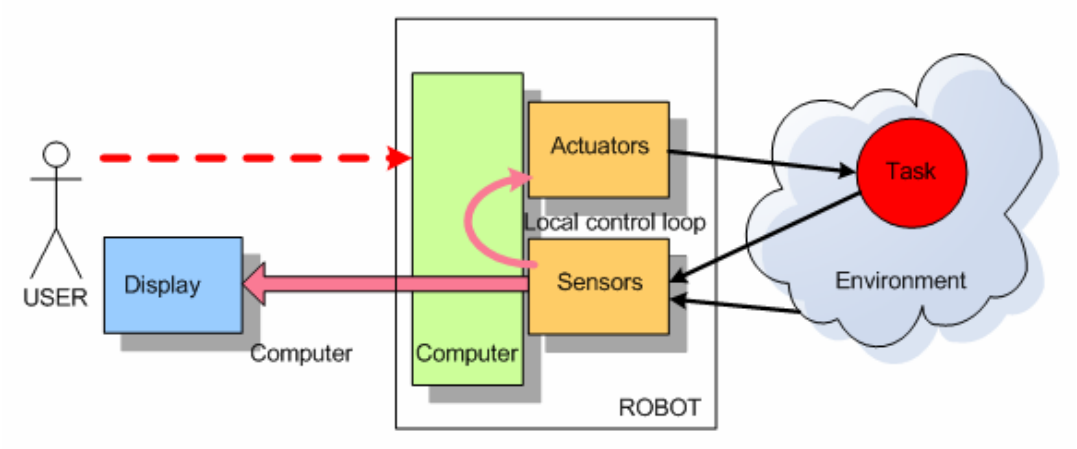


Fig. 3.4. Autonomous control scheme

Usually, for the major part of robotised tasks it is necessary to have a user being able to visualize the environment and the robot's state in order to help the robot avoid the unspecified dangerous situations. Nevertheless, totally autonomous operations are possible in the case of a well structured environment and a well determined task. One example is an industrial manipulator robot. In the case of the service robot, totally autonomous operations are not admitted. It is related to the main objective of its operation which consists of the assistance for the user in different tasks following their indications. In this case the robot is a tool which allows the user to execute different tasks. The user establishes objectives and plans the robot's actions. The local control is performed by the robot autonomously but is always supervised by a human master.

The most popular conceptual strategy for service robots is a mixed control using autonomous teleoperation. It is a kind of supervised control where the robot works in a half-autonomous mode and the operator controls upper level goals of a task execution. The user and the robot compound a complex control mechanism and collaborate in the development of a task. The role of each actor determines the functionality of a whole system. Basic questions in a functionality definition are often related to who starts the control action or "Mixed Initiative Control Methods" [Allen 99]. The distribution of control responsibilities in this system should be adjusted in accordance with functional specifications and it affects on the design of control architecture, especially on the upper level control.

3.4.2 Functional design of control architecture

A typical approach for designing a control system of a robot consists of detection of basic tasks of a robot and decompounding it into functions specifying all interactions between them. Basic sub questions that should be addressed in the stage of functional design are: the development of different levels of hierarchy of the architecture, choosing the appropriate proportion between planed and reactive functioning and defining the information about robot state and environment necessary for the control system operation [Ollero 2001].

To define the hierarchical levels of control, the descending (from the upper cognitive level to the lower actuator's control) approach is used. The bandwidth and computational complexity of every level play an important role in defining the hierarchical structure. The cycle time for realization of a function can vary from several minutes or even hours for symbolical cognitive operations, to milliseconds for the control of robot's joints.

Therefore, it is important to precisely specify all requirements of a system and estimate the complexity of each task the robot should be able to perform. Evidently, definition of hierarchical levels which doesn't include some functions or functionalities of a robot will cause the later difficulties at the moment of implementation. At the same time, the definition of levels with a lack of clear essence leads to artificial complexity of the architecture.

Figure 3.5 shows the example of a hierarchical structure with three levels of control unifying functions of perception, representation, motion and task planning, with vertical and horizontal links between them.

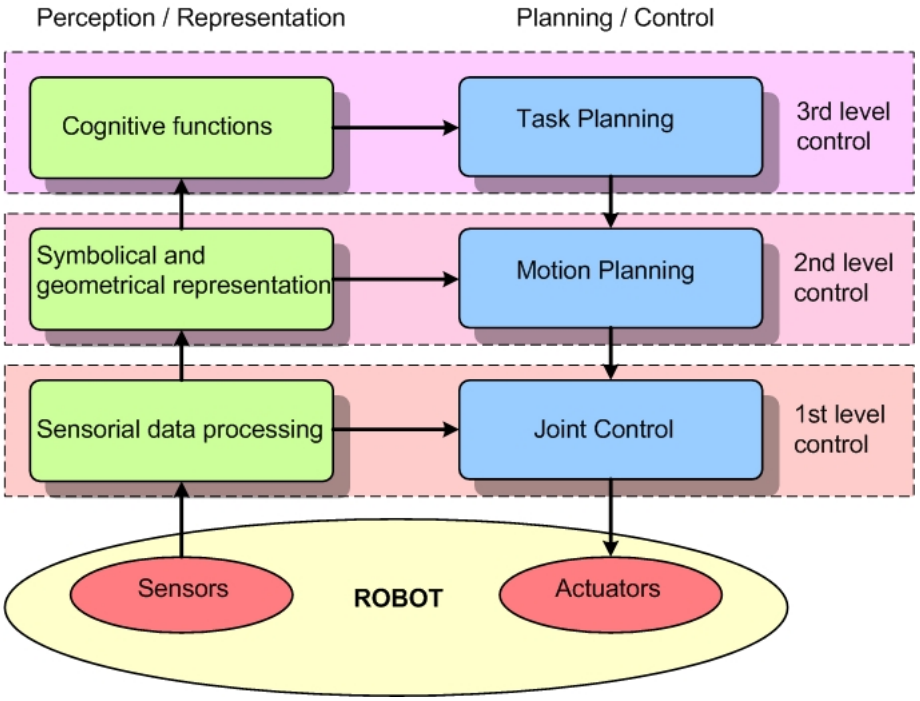


Figure 3.5: Levels of the control architecture

The representation of control architecture in the previous figure is rather general and it can vary for different types of robots. The *bottom control level* is responsible for sensorial information processing, and generation of motion reference commands for actuators. From the planning/control point of view, actions generated in this control level can respond to upper level requests or be the direct reaction of the system to input signals provided by sensors of a robot.

Also, it should be mentioned, that this control level can be divided into a smaller, hierarchical, sub-levels such as servocontrol for the joints, with a sample time of milliseconds. Trajectory generation can be located on the next, upper level with a sample time of decimal parts of a second. Also, if the control strategy allows the dynamical model consideration, it can be operated at the same sub-level.

In the first control level it is possible to install other functions allowing fast reaction of the system to an external sensorial input or internal event. In this way it is easy to organize, for example, a force/position control in a range of different applications, maintaining the end-effector of a robot (such as a special tool in the case of industrial robots, or the hand of the humanoid robot) in permanent contact with the object. Finally, this level should contain the control of the end-effector of a robot (for example open/close hand control).

Second control level is the motion planning procedure. From the perception/representation point of view, this level generates the geometrical and symbolical representation of a robot and its environment. Input information for planning is provided by the same level representation or from the upper control level. The realization of this level differs for different types of robots. For example the planning of an obstacle free route for a mobile robot differs from the planning of motion for positioning a manipulator in a space position.

This level should provide functions for motion planning taking into account the geometrical and symbolical representation of the environment and the robot's structure. Sample time for this level of control, with current hardware development, can vary from seconds to minutes and depends on the complexity of the task. Also, it is necessary to mention local real-time planning, interpreting sensorial information that is usually used for modification of previously planned motions.

This control level will also supervise bottom motion control levels. Different control strategies can be implemented at the bottom level as a function of planned motion in this, upper control level. For example, a conventional position control and high speed joint motion are available in an obstacle free planned motion, while the proximity of obstacles changes the control strategy to the slow joint motion with a position/force control and sensorial feedback.

The upper, *third level* of the control hierarchy includes the cognitive functions of a robot. In general, it can also be a compound of different sub-levels. It should be noted, that the complexity of different tasks differs radically, which means that the complexity of functions participating in cognitive planning and control also varies. Moreover, this level of control supervises the execution of motion planning functions carried out by the lower level. For example, if a route planning procedure from the previous level is not able to find any acceptable route in some determined conditions (which is a rather frequent occurrence) searching parameters can be changed, some assumptions be made, or in the worst of cases control can be passed to a human operator.

Also, another sub-level can be implemented, responsible for generating sequences of upper level orders, needed to execute the objectives of a main plan. The sample time for such control operations is measured in hundredths of seconds, in complex missions when it is necessary to fragment the task into service orders, which should also be divided into groups of sub-tasks realized on individual objects.

Another question that should be addressed during the design of functional level of control architecture is how to select the appropriate proportion between the planned and reactive functioning of a robot. Nowadays, this is a very active field of research for many robotics laboratories. The main challenge is to establish the optimal compromise between planned previously established task execution and the capacity of a robot to react on unexpected perturbations using sensorial information.

With the development of the mobile robot appears two basic approaches to the navigation control problem (it should be noted that these approaches are applicable not only for navigation of mobile robots but for general motion control of any type of robot). They are the SMPA (Sense - Model - Plan - Act) approach and the reactive approach.

The first approach is based on the explicit model of the environment and supposes that the plan is executed in the situation well known a priori. This functional architecture is based on planning with centralized models for verification of the information provided by sensors. This information is used by the plan maker in order to produce a sequence of actions. The basic disadvantage of planning based on the knowledge of the environment is the impossibility of a fast reaction given a change in the environment. Moreover, it needs a very precise and reliable model of the environment which can be very complex and expensive from the computational point of view.

The reactive architectures implement the control strategy as the collection of the condition-action operations (agents). The system consists of a collection of the reactive rules containing the internal description of each possible state. The global behaviour of the robot is the result of the interactions between different reactive agents. The reactive architecture is based on the direct linkage between sensors and actions through the fast feedback loops. This architecture works better in environments with a lot of sources of sensorial information.

Finally, as a compromise between SMPA and reactive architectures there is a hybrid architecture that is the combination of these two presented above. It includes a reactive system for the bottom level control and a planning module for decision making at the upper level of control. The entire control system is separated into two or more parts with communications between them but basically independent from each other. Usually it is possible to make sufficiently precise models for well structured static environments. This allows implementation of a control system mainly oriented on the execution of a main plan and having some capacities to react on the unexpected perturbations. In general, the reactive process of the bottom part of the control architecture is engaged in providing safety for the robot while the upper level process uses the planner in order to select the appropriate future action.

In order to define the information about robot state and environment in real time the control system should make a representation of it using some kind of maps and symbolical representations in special data structures. For example, the representation of the environment by a vision system includes different data such as images, geometrical and symbolical objects. In spite of the fact that the environment where the robot operates is basically static with relatively

small changes, the definition of the state of a robot and structures to represent this state and environment affect on the planning efficiency and the reliability of the system.

Also, it is important to add to the functional definition of the control architecture of a robotics system the definition of priorities (related to the security reasons) and assignation of different tasks.

3.4.3 Physical design. Topology of a control system

As it was shown in the previous chapter, there are a lot of types of robots which are widely used in different fields. The design of a control system depends on functional specifications provided by a user or a task. Physical design tries to define the relation between system functions and physical resources – hardware and software. A robots' control architecture consists of a number of microprocessor-based modules that perform different pre-programmed tasks. Some of them are connected to the environment (sensors/actuators) and some perform internal functions such as data storage/processing or organizing parts of the robot's architecture to act as one system.

General requirements for a control architecture design presented in a previous section provide some guidelines for the future physical topology of hardware and software systems of a robot. Main considerations concerning the physical design of a control system are: the level of modularity of the architecture and degree of distribution/centralization of components and functions.

Level of Modularity

The modularity of the architecture refers to the design of a system containing separated components that can be connected together. The main advantage of modular architecture is that it is possible to replace or add any one component (module) without affecting the rest of the system. The opposite of a modular architecture is an integrated architecture. In this type of architecture no clear division between components exists, but only a single system performing different operations. Figure 3.6 presents block-schemes of modular and integrated architectures.

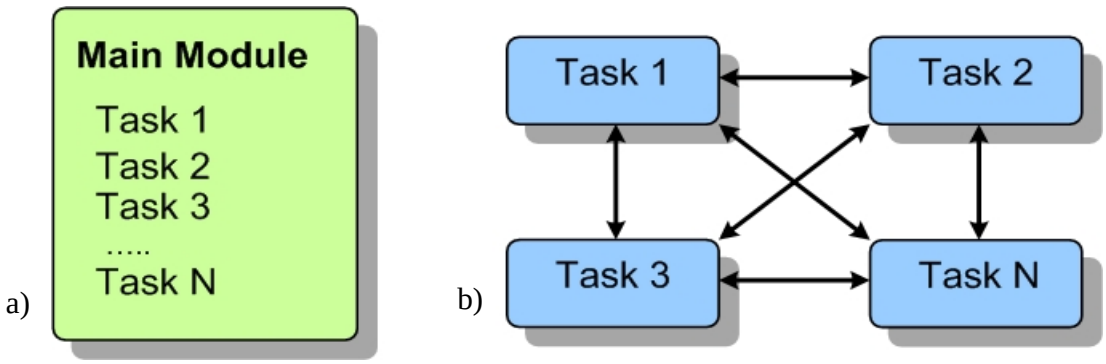


Fig. 3.6: Modularity level of control a) Integrated architecture b) Modular architecture

The term "modular" can be applied to both, hardware and software components of control architectures. Modular software design, for example, refers to a design strategy in which a system is composed of relatively small and autonomous routines that fit together. Modular

hardware components can be described as small modules responsible for a special control task as in the example of a driver controlling the motion of one motor. Integrated architecture refers to two or more components merged together into a single system. For example, any software component that performs more than one task can be described as integrated.

Creating appropriate modular architecture is a complex task which requires the precise knowledge of functionality of a system and its application. The modularity of architecture has a two-part definition. The first part is a decomposition of the overall functionality of a system into a set of defined functions and component parts that are going to provide those functions. The second part of the definition is the specification of the interface between the components, in other words, how components will interact together in a system. The specification of the interface is critical to the design of flexible architectures that allow substitution of component variations within an architecture without having to make adjustments in other components. For example, the architecture of most PCs allows us to easily replace an older hard disk drive with a new one, because the interfaces between the hard disk drive and the rest of the computer have been defined to allow a range of variations in hard disk drives.

As the modularity does not suppose that all modules should be of the same level, in addition to the completely modular and completely integral architectures there is a hybrid architecture that is a combination of modularity at a bottom level with the integral base as the control kernel of a system. Small control modules execute their tasks as for example in sensorial data processing or communication. The integral control module performs a multitude of tasks such as supervising the execution of all bottom level tasks, general motion planning, sending commands of the upper control level, etc.

Degree of distribution/centralization

This leads to another aspect of the physical design of control architecture – the degree of distribution of components and functions. On-board control hardware and software meet the requirements of physical topology and a task. From this point of view the control architectures can be classified as centralized and distributed architectures. Figure 3.7 shows the realisation of these control architectures.

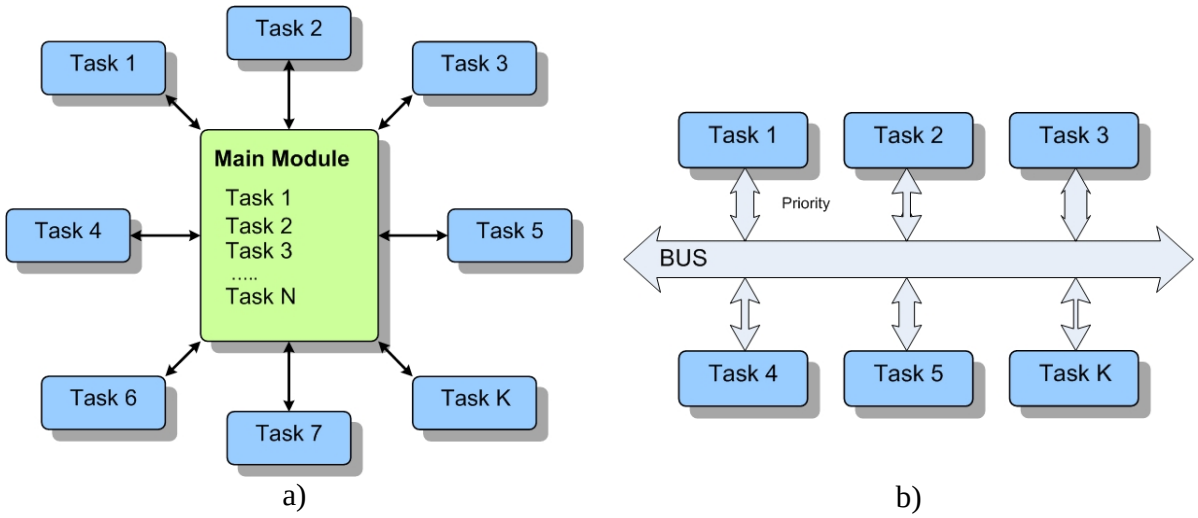


Fig. 3.7: Centralization level of control. a) Centralized architecture b) Distributes architecture

Usually, a centralized control system has a hierarchical structure in which different modules have different priorities. When a robot's brain gives a command to its arm to grasp an object, it doesn't care how an arm should move its fingers. It is logical to close the feedback between the fingertip touch sensors and the finger motors on the arm level providing the brain with a high-level grasp function. So, it means that each module of the robot should have its master (except the topmost, in most cases) and, probably, some slaves. And during normal operation they should not address any module except their immediate superior or subordinates.

In another words, centralized control architecture unifies various modules dedicated to distinct tasks such as perception, task planning, motion planning and motion control. All these modules communicate with a central control module which holds information about the robot's actual state and environment. This central module is also responsible for providing communication and synchronization between every part of the system. Figure 3.7(a) shows the realisation of this kind of architecture in the case of software modules. Modules are joined by a data flows.

In the case of a hardware centralized system the organization of interaction between principle modules will be practically the same. Each hardware control is internally based on wired connections, allowing each module to interact with a single master one level above and several (if any are required) modules one level below. Wireless communication is used primarily for inter-robot communication, which does not actually need to be between the robot's central modules - it is possible to link wirelessly some lower-level robots' parts.

From figure 3.7(a) it can be clearly observed that the centralized architecture supposes hybrid modularization. The central master module performs multiple tasks that lead to the integral structure while the bottom level blocks represent a typical modular structure.

On the other hand, another approach is to distribute tasks and functions between modules. Thus, every module will have the same level but different priorities. This control architecture is shown in the figure 3.7(b). There are a lot of different modules in connection with each other in compliance with their functions and tasks. The main objective is to determine the priority. The module with higher priority will be utilised more frequently or with shorter delays when waiting for its turn.

In the case of hardware distributed control systems, the role of communication is even greater then in the centralized control architecture. As all processes are distributed, all internal information will circulate in a communication bus. In this context, one of the basic rules for the implementation of a distributed control system is: "what is needed to be fast should be local" because the links (especially wireless ones) have limited bandwidth.

Communication links in robot control have two distinct functions: data exchange and synchronization of internal clocks of all subsystems. One problem in such systems is the synchronization between modules when a precisely coordinated activity involving different parts of the same robot, or even different robots, is needed. These synchronization functions will not just synchronize the system's clock, timing different modules, but also provide competitive procedures for accessing the communication bus.

Finally, when we speak about centralized or distributed architectures we refer to both hardware and software systems. Nevertheless, systems can exist having a mixed architecture. For example centralized hardware with distributed software modules or vice versa, centralized software system with distributed hardware. In general, the level of centralization/distribution of

hardware and software may vary and depends on the main task of the robot. It should be mentioned that the implementation of the centralized architecture presented in a figure 3.7(a) is more suitable for task planning control strategies while the implementation of distributed architecture is preferable for reactive control strategies.

3.5 The concept of a motion in robotics

Up to this point, only general considerations of robot control architecture have been made. However, the topic of the thesis is the design of motion control architecture (the word "motion" is the key). First of all, let us introduce the fundamentals of a motion concept in robotics and then extend it to particular cases of humanoid robots.

Functionally, a robot can be considered a physical agent, capable of action for the achievement of tasks specified by a human. Action can be characterized by moving in space, object manipulation and interaction with other beings. To perform all these tasks the robot should have different systems and abilities:

Mechanical structure - As any robot is an artificial body, it is a mechanism which has a series of mechanical linkages. The main purpose of the mechanism is to establish kinematics constraints and to deliver the motion of the robot to the specified point. In general the mechanism consists of joints and links. A link is a rigid body inside a mechanism. A joint is the point of intersection between any pair of adjacent links.

Kinematics - Changing the relative position of the robot's links we can produce different types of motions. The relationships between the motion parameters of the robot's link and the particular point of the mechanism are expressed in two robot kinematics problems. A forward kinematics problem is formulated as: how to determine the location of the robot's specified point if joint positions are known. An inverse kinematics problem can be formulated as: how to determine the joint positions necessary to move the mechanism to the specified point.

Actuators – Actuators are used to produce the forces and torques necessary to set in motion the mechanical structure of the robot. Nowadays, the most widely used source of energy is electrical energy. Therefore, electrical motors (particularly DC motors) are widely used in robotics. Nevertheless, there are a lot of works on other types of actuators. For example, applying biomechanics concepts, artificial muscles have been developed. Although this can't replace conventional electric actuators because of its limited power efficiency, in the future it is expected to play a more important role in robotics.

Dynamics – The robot is a machine which executes motion. Dynamics estimates the relationship between motion parameters and forces/torques produced by actuators. Also it includes concepts related to mass distribution and inertia of the rigid body.

Sensorial system – Any engineering system working in a real world always suffers certain noise and disturbances. Therefore, the actual state of inputs and outputs of the system don't coincide with the ideal ones. The only way to obtain the real state of the system is sensing, three different types of which can be considered. The first type of sensing is actuator's motion data, which is used for the feedback. This group of sensors includes different types of encoders and proximity sensors operating with a single joint of the robot. The second type of sensorial system is used for the measurement of the interaction of the robot with the environment. Different types

of force/torque sensors, gyros, accelerometers and inclinometers are usually used in order to determine how the environment affects or changes robot behaviour and then react to these actions or changes. And finally, the third type of sensorial system is used to determine motion parameters of the environment or working pieces the robot is working with. This kind of sensing uses a camera, laser and sonar systems and so on to acquire data from the environment and then plan action accordingly.

Control – Robots are designed to perform tasks, through motion. As the task is usually defined as some kind of constraint, the dynamics of the robot should strictly follow the dynamics imposed by the task. The feedback control is a phenomenon whereby some proportion of the output signal of a system is passed (fed back) to the input. How to determine the desired dynamics for a given task is considered in another of the robot's aspects – the motion planning.

Information system – As human beings have a brain to conduct their behaviour, the robot should also have one system which supervises the others and aims to ensure the goal s achieved. From a hardware point of view, the robot's brain can be considered to be the microcontroller, or CPU, having a certain level of computational capacity. In terms of software, the brain is a set of applications which govern all hardware and offer a set of solutions for successfully achieving the goal. The first function of the information system is to perform computational and communicational tasks, such as sensory-data processing and mathematical computation of the kinematics and dynamics, as well as the control of the robot. The second function is communication between the robot and the environment. The third, most advanced function of the robot's brain, is the development of artificial intelligence – the ability to plan goals without the human master.

All the above concepts have a unique purpose – to provide the robot with the possibility to act as a physical agent in the environment. The action in the environment means to produce motions. Finally, the motion, being the visible form of action, may be then considered as the main unifying concept in robotics (Fig. 3.8).

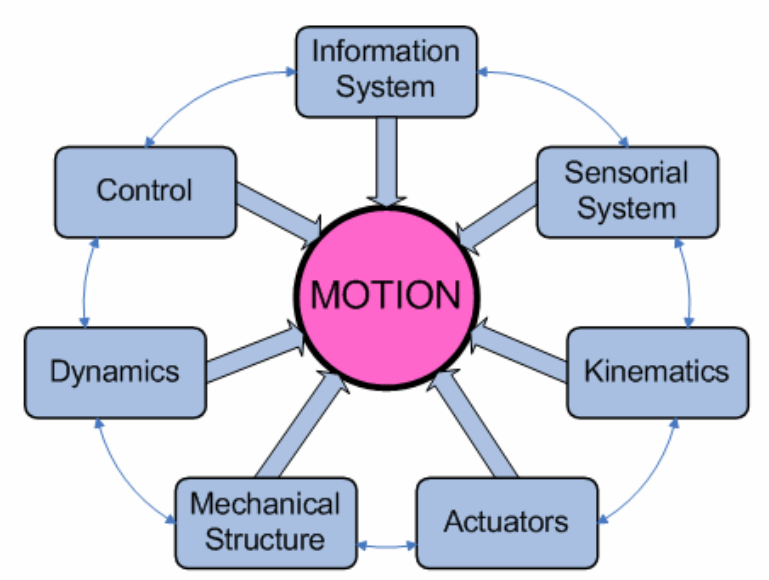


Fig. 3.8: Motion centered robotics system

As motion is a key concept in robot design, the motion control architecture, integrating the major part of the abilities presented below, can be considered as its essential element. Referring to the functional design of robot control architecture in general (figure 3.5), motion control is its lowest control level. The realization of the motion control architecture depends on the type of the robot to be designed, but the most common concept is the control of robot's joint motions. The following section provides a short review of the conventional methods of joint motion control for robotics systems.

3.6 Joint motion control

The main goal of every motion control system is to provide a task achievement by the executing of a motion. Motion usually is provided by an actuator (usually an electric motor). An actuator is a device that uses electrical energy to produce mechanical energy. This energy produces a mechanical movement which consists of motion of the joint of a robot. For the stable motion of a robot, joint motion should be precise and reliable. Therefore, every motion control system at its bottom level is provided by the servo control. Let us make a short study of the joint motion control in classical control theory manner. The use of more complex intelligent control techniques (like neural networks, Bayesian probability, fuzzy logic, machine learning, evolutionary computation and genetic algorithms) is not so frequent in the area of motion control and, therefore, these algorithms are not part of this review.

The fundamental concepts of servo motion control have not changed significantly in the last 50 years. The need to improve transient response times, reduce the steady state errors and reduce the sensitivity of a joint to load parameters is the basic reason for using servo systems in contrast to open loop systems.

Improving the transient response time generally means increasing the system bandwidth. Faster response times mean quicker settling allowing for higher system's productivity. Reducing the steady state errors relates to servo system's accuracy. Finally, reducing the sensitivity to load parameters means the servo system can tolerate fluctuations in both input and output parameters.

Servo control in general can be divided into two fundamental classes of problem. The first class deals with command tracking. It addresses the question of how well the actual motion follows what is being commanded. The typical commands in rotary motion control are position, velocity, acceleration and torque. For linear motion, force is used instead of torque. The part of servo control that directly deals with this is often referred to as "Feedforward" control. It can be thought of in terms of what internal commands are needed so that the user's motion commands are followed without any error. Feedforward control assumes that a sufficiently accurate model of both the motor and load is known.

The second general class of servo control addresses the disturbance rejection characteristics of the system. The disturbances in motion control systems can be anything. It may be torque disturbances on the motor shaft or incorrect motor parameter estimations used in the feedforward control. The "P.I.D." (Proportional Integral and Derivative position loop) and "P.I.V." (Proportional position loop Integral and proportional Velocity loop) controls are used to combat these types of problems. In contrast to feedforward control, which predicts the needed internal commands for zero following error, disturbance rejection control reacts to unknown disturbances

and modelling errors. Complete servo control systems combine both these types of servo control to provide the best overall performance [Kaiser 2001].

Traditional PID and PIV controllers are generally good enough for most motion control applications. The ubiquitous proportional-integral-derivative controller is especially cheap and easy to implement. Tuning PID and PIV controllers is a relatively straightforward operation that can be accomplished with a few empirical tests.

However, even with these enhancements a PID based controller leaves considerable space for improvement. Once tuned, it can only control the process it started with. If the behaviour of the process changes appreciably after start-up, the controller may no longer be able to counteract the error when a load disturbs the process variable. If the mismatch between the process behaviour and the controller's original tuning becomes very strong, the closed-loop system may even become unstable. The traditional way to fix this problem is to start again and make a retune of the control loop whenever its performance degrades. This is not difficult, but repeatedly tuning and retuning a loop is tiring and time consuming. In some cases, manual retuning may not even be possible if the behaviour of the process changes too frequently, too rapidly, or too much. Therefore, the implementation of the on-line adaptation algorithm is needed [VanDoren 2003]. One of the most compelling reasons to replace or provide the PID based loops with adaptive control loops is convenience. A controller that can continuously adapt itself to the current behaviour of the process relieves the need for manual tuning.

The following sections provide a brief introduction to the basic joint control techniques used in contemporary motion control architectures.

3.6.1 PID

The basic components of a typical servo motion system are showed in the figure 3.9 using standard Laplace notation. In this figure, the servo drive closes a current loop and is modelled simply as a linear transfer function $G(s)$. Of course the servo drive has peak current limits, so this linear model is not entirely accurate. However it provides a reasonable representation for the analysis. Basically, the servo drives receive a voltage command that represents a desired motor current. Motor shaft torque T is related to motor current I, by the torque constant K_t. Equation (3.1) shows this relationship:

$$T \approx K_t \cdot I \tag{3.1}$$

The transfer function of the current regulator or torque regulator can be approximated as unity for the relatively lower motion frequencies we are interested in. Therefore it is possible to make the following approximation shown in (3.2).

$$G(s) \approx 1 \tag{3.2}$$

The servomotor is modelled as a lump inertia J, a viscous damping term b, and a torque constant K_t. The lump inertia term is comprised of both the servomotor and load inertia. It is also assumed that the load is rigidly coupled so that the torsional rigidity moves the natural mechanical resonance point well out beyond the servo controller's bandwidth. This assumption

allows us to model the total system inertia as the sum of the motor and load inertia for the frequencies we can control. Somewhat more complicated models are needed if coupler dynamics are incorporated. A more complete discussion on this topic will be provided in the next chapter.

The actual motor position $\theta(s)$ is usually measured by either an encoder or resolver coupled directly to the axis of the motor. Another important assumption is that the feedback device is rigidly mounted. Thus, mechanical resonant frequencies can be ignored. External shaft torque disturbances T_d are added to the torque generated by the motor's current to give the torque availability to accelerate the total inertia J [Kaiser 2001].

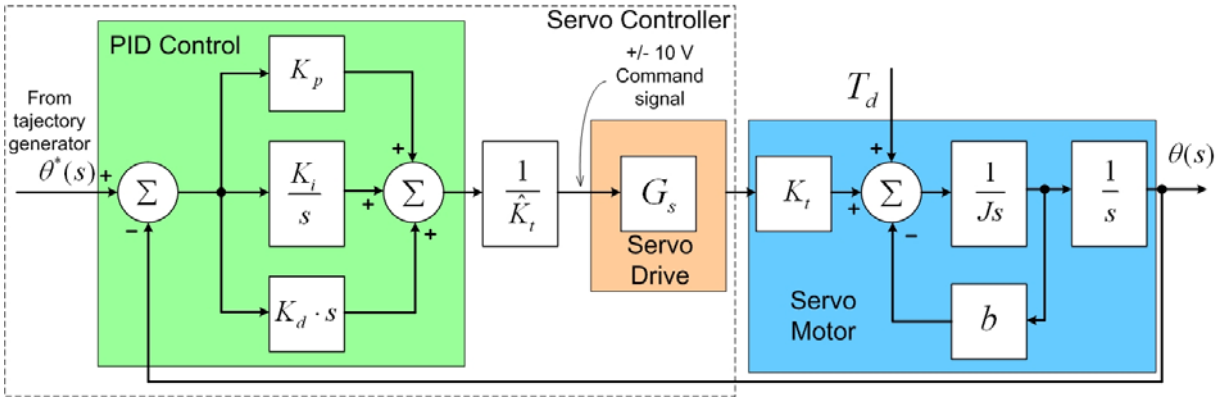


Fig. 3.9: Basic PID servo control topology [Kaiser 2001]

The servo controller closes the position loop and is located around the servo drive and motor block. A basic servo controller generally contains both a trajectory generator and a PID controller. The trajectory generator typically provides only a set of position commands labelled in figure 3.9 as $\theta^*(s)$. The PID controller operates on the position error and a torque command is its output. This output can sometimes be scaled by an estimate of the motor's torque constant labelled as $\hat{K}_t$. If the motor's torque constant is not known, the PID gains are simply re-scaled accordingly. The symbol "^" is used to indicate that it is a value, estimated in the controller because the exact value of the motor's torque constant is generally unknown. In general, equation (3.3) holds with sufficient accuracy so that the output of the servo controller (usually + / - 10 volts) will command the correct amount of current for a desired torque.

$$\hat{K}_t \approx K_t \tag{3.3}$$

There are three gains (proportional, integral and derivative) to adjust in the PID controller K_p, K_i and K_d. All these gains act on the position error which was defined in (3.4). The superscript "*" refers to a command value received from the trajectory generator.

$$error(t) = \theta^*(t) - \theta(\mathrm{t}) \tag{3.4}$$

The output of the PID controller is a torque signal. Its mathematical expression in the time domain is shown in the equation (3.5):

$$PIDoutput(t) = K_p\big(error(t)\big) + K_i \int \big(error(t)\big)dt + K_d \frac{d}{dt}\big(error(t)\big) \tag{3.5}$$

There are two basic ways to operate for selecting the adequate PID gains. The first one is a trial and error method. The second one is an analytical approach. The use of a trial and error approach relies significantly on the designer's own prior experience with other servo systems. The one significant imperfection of this is that there is no physical insight into what the gains mean. Further more, there is no way to know if the gains selected by a designer are optimum by any definition. However, for a long time this was the approach most commonly used in the area of automatic control. In fact today it is still systems with required low performance which are usually found in process control.

Nevertheless, an analytical approach to tune controller's gains also was created. Ziegler and Nichols proposed a method based on their experience in industrial control. Although they originally intended their tuning method for use in process control, their technique can be applied to servo control as well [Ziegler 42].

Generally speaking, it is based on the influence of PID gains on the system's response. The proportional term affects the overall response of the system to a position error. The integral term is needed to force the steady state position error to zero for a constant position command. The derivative term is needed to provide some damping action, as the response becomes oscillatory. However, all these three parameters are strongly related so that the adjusting of one parameter will affect any of the previous parameter adjustments. Figure 3.10 shows the influence of K_p, K_i and K_d gains on the step response of a system.

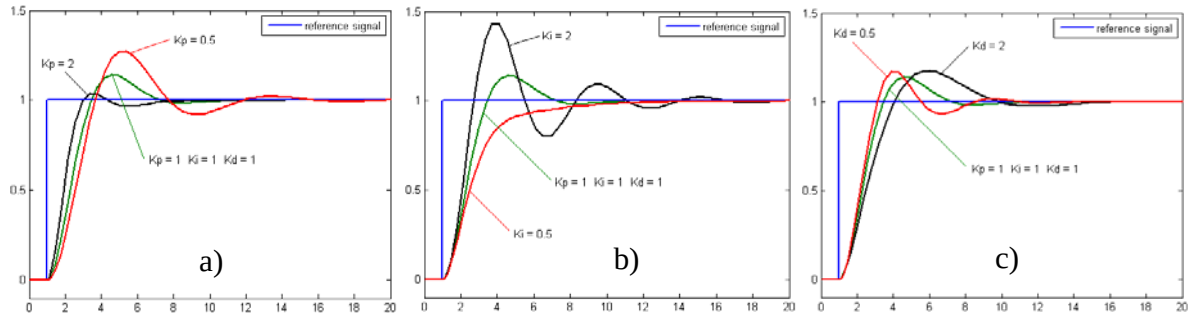


Fig. 3.10: Change of response for varying a) K_p b) K_i c) K_d

Some applications may require the use of only one or two modes of error regulation to provide the appropriate system control response. This can be achieved by setting the gain of undesired control outputs to zero. In such a case PID controllers are called PI, PD, P or I controllers. For example PI controllers are particularly common. Derivative action is very sensitive to measurement noise, thus the absence of an integral part prevents the system from reaching its target value due to the control action.

3.6.2 PIV

For better prediction of system response some alternative topology was proposed. One example of an easier to tune topology is the PIV controller shown in figure 3.11. Basically this controller combines a position loop with a velocity loop. More specifically, the result of the position error multiplied by the proportional gain K_p becomes a command for the velocity correction. The integral gain K_i now operates directly with the velocity error instead of the position error as in the case of the conventional PID controller. Finally, the K_d term in the PID position loop is replaced by a K_v term in the PIV velocity loop [Kaiser 2001].

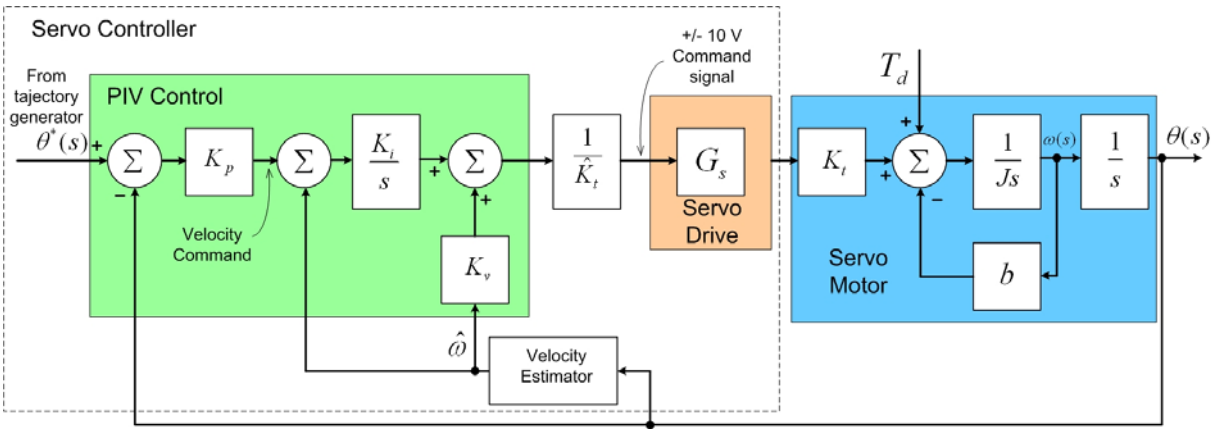


Fig. 3.11: Basic PIV servo control topology [Kaiser 2001]

Besides the actual position, the PIV control requires the knowledge of the motor velocity. It is provided by a velocity estimator showed in figure 3.11. This usually can be a simple filter. However, significant delays can appear and it should be taken into account if really accurate responses are needed. Alternatively, the velocity can be obtained by use of a velocity observer. In either case, a clean velocity signal must be provided for PIV control.

To tune this kind of system, only two control parameters are required. They are the bandwidth (BW) and the damping ratio (ξ). With the fixed damping ratio, the bandwidth directly relates to the system rise time. The higher is the bandwidth, the quicker the rise and settling times are. The structure of the PIV control allows it to reject unknown disturbances affecting the system. In general, if the disturbance rejection is high, the system is stiffer and more likely to provide repeatable moves in the presence of unknown disturbances.

3.6.3 Feedforward

Feedforward control can be employed in order to achieve near zero tracking error. In contrast to a feedback system, a system which manifests feed-forward behaviour responds to a measured disturbance in a pre-defined way.

A basic requirement for the design of feedforward control is the availability of both the velocity, $\omega^*(s)$ and acceleration $\alpha^*(s)$ commands synchronized with the position commands

$\theta^*(s)$. An example of how feedforward control is used in addition to disturbance rejection control is shown in figure 3.12.

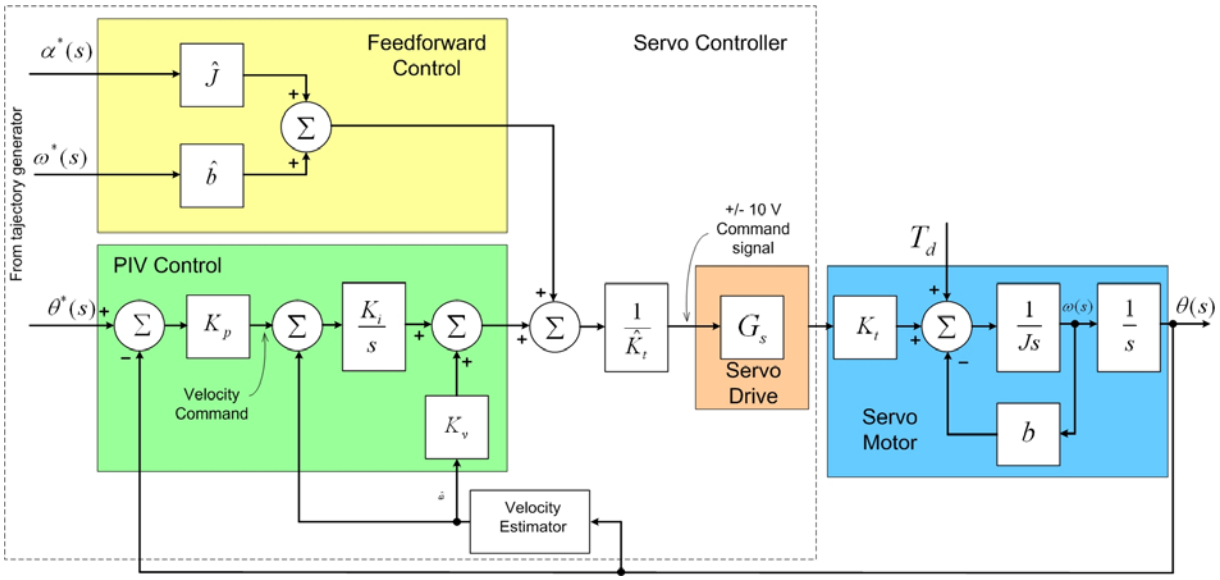


Fig. 3.12: Basic feedforward servo control topology [Kaiser 2001]

Feedforward control operates calculating the required torque needed to make the desired movement. The dynamic equation of motion in terms of the motor torque is shown in equation (3.6):

$$T_{motor} - T_d = J\alpha + b\omega \tag{3.6}$$

As the disturbance torque T_d is unknown, the estimated motor torque can be approximated as is shown in equation (3.7):

$$T_{estimated}(s) = \hat{J}\alpha^*(s) + \hat{b}\omega^*(s) \tag{3.7}$$

In most cases, the disturbance torque is small enough that the estimated torque is very near the required torque. In this case, if the velocity and acceleration commands are known (estimated), simple estimation of the total inertia and viscous damping can be used to generate the estimated torque profile in real time.

When there is a disturbance that can be measured before it affects the output of the process, combined feedforward and feedback controls can significantly improve the performance of the system over simple feedback control. In ideal situations, the feedforward control can eliminate the effect of the measured disturbance on the process output. Feedforward control can reduce the effect of the measured disturbance on the output better than can be achieved by feedback control alone, even when there are some modelling errors. However, the decision on whether or not to use feedforward control depends on whether the degree of improvement in the response of the system justifies the added costs of implementation and maintenance.

To summarise, disturbance rejection control can be obtained in one of a number of ways, the two most common are PID and PIV controls. The direct use of PID control can often meet low performance motion control loops and are generally set by either the Ziegler Nichols or by trial and error methods. Overshoot and rise time are interconnected making gain adjustments difficult. PIV control on the other hand, provides a method significantly decoupling overshoot and rise time, allowing very high disturbance rejection characteristics and relatively easy design. Finally, feedforward control can be added to disturbance rejection control in order to minimize the tracking error.

3.6.4 Adaptive Control

As mentioned above, if the mismatch between the controlling process and the controller's tuned parameters becomes very strong (it may occur because of changes in internal dynamics of the process or strong external disturbances), the control system may produce big errors or even become unstable. In these cases the implementation of the on-line adaptation algorithm is needed.

Adaptive control modifies the control law used by a controller in order to adjust it with the knowledge that the parameters of the system under control may be time-varying or uncertain.

There are several broad categories of linear adaptive control, but all fall into one of three basic categories [Astrom 94]:

- Gain scheduling (figure 3.13).

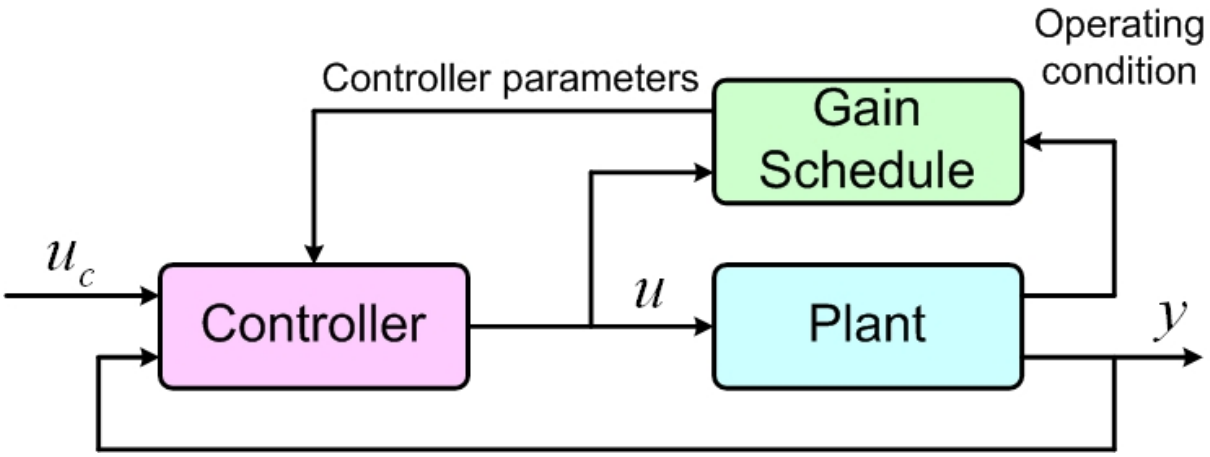


Fig. 3.13: Gain scheduling adaptive control

Usually it is possible to equip a traditional controller with a different set of tuning parameters for each possible value of the set point, but each set has to be previously manually adjusted. After that, the adaptive controller can perform automatically applying tuned controller parameters according to system behaviour and operating conditions. An advantage of these adaptive algorithms is that they are very fast and direct (no complex online calculations are needed). On the other hand, quite a lot of design work before starting the control is needed.

- Model Reference Adaptive Controllers (MRACs) (figure 3.14).

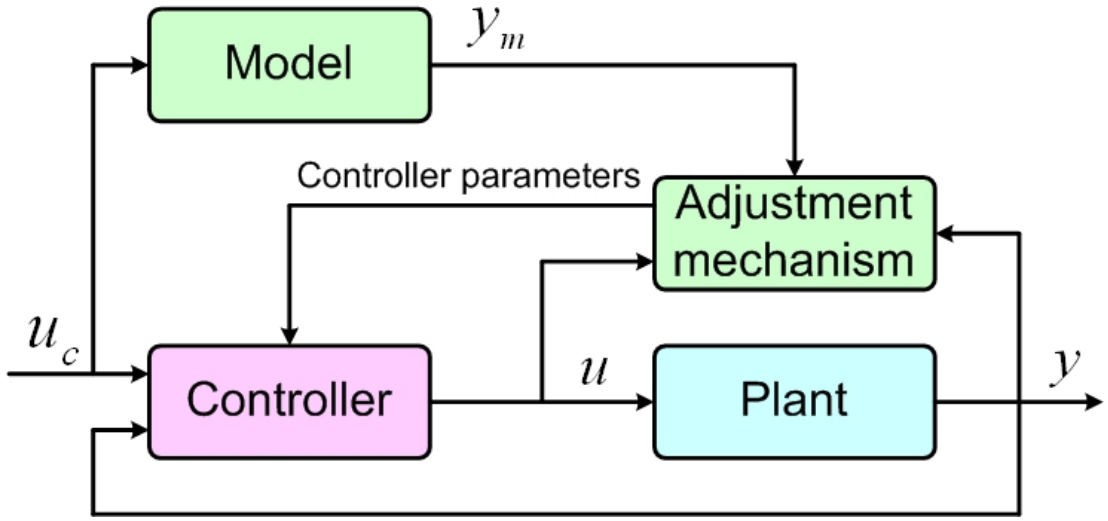


Fig. 3.14: MRAC adaptive control

These control methods incorporate a reference mathematical model of the process defining desired closed loop performance. The hard part of model-based adaptive control technology is generating or identifying the model. Given a model of the process, it is relatively easy for a controller to design an effective control law. After all, an accurate model that can correctly predict the future effects of current control efforts contains all the mathematical information that the controller needs to select its control actions so as to produce the desired process outputs in the future.

- Model Identification (Self Tuning) Adaptive Controllers (MIACs) (figure 3.15).

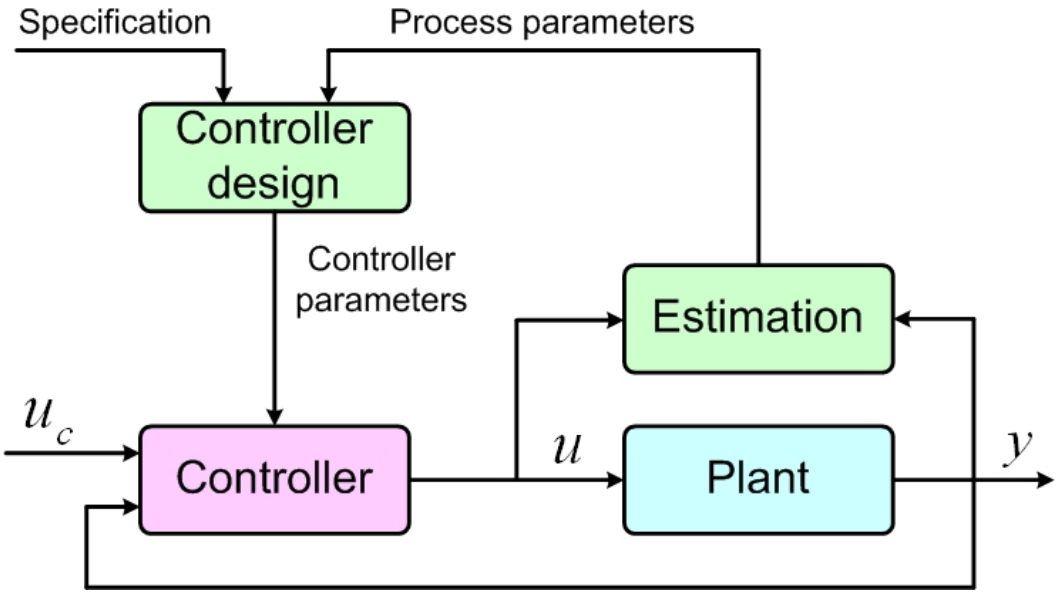


Fig. 3.15: Self Tuning adaptive control

These control methods perform system identification while the system is running. These techniques are universally considered to be the best approach to adaptive control. The control algorithm doesn't require pre-knowledge of system parameters and disturbances. However, this adaptive control method requires implementation of complex systems that leads to a complex analysis and design.

Section 3.6 presented basic concepts of the classical mono joint motion control. All controllers presented in this section are linear. Thus, their performance in non-linear systems is variable. Often these controllers are enhanced through methods such as adaptive control techniques. More detailed consideration of a joint control problem will be presented in chapter 4.

3.7 Case study

Before addressing motion control architecture for humanoid robots, let us examine a case study of motion control architectures of the basic types of contemporary robotics systems presented above. It will provide us with knowledge of modern tendencies in robotics and will allow for the design of new hybrid types of architectures for humanoids, adopting appropriate modules and principles from the conventional architecture types. Here, we will consider three basic types of motion control architectures - polyarticulated, mobile and humanoid systems. This study will consider only the basic elements of the motion control architecture – typical software and hardware realizations and special control features. Afterwards, a discussion will be presented.

3.7.1 Control architectures for polyarticulated robots.

Considering the motion control architecture of the polyarticulated robot, we will consider mainly industrial manipulator types of robot because it is the most common robot in this group. An industrial robot is a programmable, multi-function manipulator designed to automate tasks such as welding or the movement of materials through variable programmed motions. Robots are capable of performing a wide variety of tasks and are an integral part of automated manufacturing systems.

Industrial robots consist of a number of rigid links connected by joints and controlled by a computer. The link assembly, or robot arm, is connected to a body which is mounted onto a base. A wrist attached to the robot arm uses an end effector to facilitate gripping or handling. The complete motion of the end effector is accomplished through a series of motions and positions of the links, joints, and wrist.

Generally speaking, it is possible to consider the industrial robot's intelligence as the capacity to move its specified point of the end effector, usually a TCP (Tool Centre Point), into a specified position inside the work space. The mission of the motion control system in this case is to guarantee kinematics and dynamics control of the motion and to process digital and analogical inputs and outputs in order to provide interaction with other devices (sensors, PLC, other robots) in the same work space.

Different motions are specified in the motion program indicating a series of goal points for TCP, logical sequences and some other data such as the type of the motion, velocity and precision of the working trajectory. There are many different programming languages for

industrial manipulators such as VAL, RAPID, and MELFA BASIC IV. Each manufacturer of industrial manipulators tries to specify its own programming language and the field in general is proprietary and closed. This doesn't allow for the standardization and unification of industrial robot control systems at the motion control level.

Besides the user program, there are two general control tasks inside the controller. They are the motion planning and the control. Motion planning is based on the kinematics model of the manipulator. It allows for computing of joints coordinates and velocities at every point of the trajectory. Usually, the controller of the manipulator is working in PTP (Point To Point) mode with linear or circular interpolation. PTP mode is the motion between two points of the trajectory in the smallest time. By the path control (CP) the robot moves between mathematically defined trajectories (straight lines or arcs between two points for example). Programmed goal points in both cases can be approximate. The major part of industrial applications supposes the displacement of the robot's TCP in Cartesian space following some predefined trajectory. Sometimes this is necessary to maintain the desired orientation of the tool carried by the robot.

As mentioned above, a unique model describing the control architecture of the poly articulated robot doesn't exist because of its variety and in the case of industrial robots because of the use of proprietary standards in data processing and communications. However, usually the control architecture consists of the following basic elements:

- Main computing system. Usually it is a CPU processing program data. It allows the creation, and execution of the program.
- EPROM (Erasable Programmable Read-Only Memory) for storing the control program, RWM (Read Write Memory) memory which is used for user program storing and RAM (Random Access Memory) memory used for storing operative data during the user program execution.
- Servo controllers or drivers for controlling the motor dynamics of each joint. Its number depends on the number of degrees of freedom the robot has. Also there may be implemented a multiaxis controller, controlling all joints of the robot.
- Amplifiers providing sufficient power for the motor. Also they can be integrated into a servo controller.
- Analogue and digital Input/Output (24VDC) modules. All I/O is divided into dedicated signals (for internal use in buttons and control indicators) and process signals which can be used for the communication of the robot with external machinery. Usually all industrial robots have a basic I/O module and the possibility to increase the number of I/Os in accordance with application requirements.
- Input and control terminal which is used for manual robot's joint motion and programming.
- External communication interface providing the connection of the robot with the external computer
- Control workstation which is used for PC control and programming of the manipulator.

The presented architecture describes the industrial robot functioning in well known and structured environments. In the unstructured or changing environment the architecture should

contain other parts providing trajectory generation and obstacles avoidance. This architecture is more complex and lies outside of the motion control topic.

3.7.2 Control architectures for mobile robots.

Mobile robots have the capability to move around in their environment and are not fixed to one physical location. Mobile robots are the focus of a great deal of current research and almost every major university has one or more labs that focus on mobile robot research. Mobile robots are also found in industry, military and security environments. They also appear as consumer products, for entertainment or to perform certain tasks like vacuum cleaning or mowing.

Mobile robots carry out many varied tasks. During these tasks the robot moves and orients. While navigating, it uses signals from the environment and the contents of its own memory to make the correct decisions. This form of navigation may be manifold depending on the given task and problem. Often the goal can be sensed, there is no obstacle between the goal and the robot. However there are numerous times when this is not the case, then the marking points must be sensed and the route known. In order for the robot to be able to do this, its motion control architecture should contain two main components:

- Drive, motion
- Control, direction

Mobile robots change their position in space while carrying out their tasks. The most common is a two-dimensional (2D) change of position. Our environment in reality is complex and not structured. In this environment the robot has to move, navigate and overcome the posing difficulties. The designer has limited and uncertain data about the surroundings, and during the robot's motion it has to deal with many unexpected occurrences and problems [Matijevics 2007]. Therefore, the motion control problem of a mobile robot is mostly a navigation problem. One possible way to create navigation is to apply artificial intelligence techniques in which the connection with the environment may be put into several sub-categories or tasks. Therefore the software architecture of mobile robots usually has a hierarchical structure [Ollero 94].

On a hardware level the navigation problem of mobile robots is solved with micro controllers and microprocessors. These micro devices have different capacities, they handle more or less in and outgoing signals and have various memory capacities and operating speeds. The tasks of the robots may vary significantly from one to another; they range from fairly simple tasks to the execution of quite complicated tasks. This fact determines the architecture and complexity used for the robot's navigation. Within a mobile robot the architecture applied may be divided in the following way:

- centralized architecture for the execution of all tasks
- one-level distributed architecture with several micro controllers, using industrial bus

In conclusion, it should be noted that the motion ability of a mobile robot is restricted by wheel displacement and is not a very difficult problem from the motion control point of view. The navigation and intelligence studies particular to this type of robot are not the topic of this thesis.

3.7.3 Control architecture of humanoid robots.

The main objective of the thesis is to propose and design novel advanced control architecture for humanoid robots. Logically, it is necessary to study already existing systems. The area of humanoid robots is still limited by the very high cost of maintenance and development. Also, as it is a high-tech area and in the vanguard of contemporary technology, it is rather inaccessible. The interchange of technologies and knowledge is very restricted therefore making a case study of the control architecture of the modern humanoid robot difficult. Nevertheless, limited data which can be found from open sources of information will be presented in this section and analysed. Later this case study will be taken as a starting point to propose original control architecture in following chapters.

The main task of control architecture for humanoid robots is to provide them with a bipedal locomotion. How this task should be realized on a conceptual level (i.e. control algorithms and procedures needed to organize a stable bipedal locomotion) will be presented in a following chapter. Now, let us present only basic hardware and software solutions for more relevant humanoid robots.

As it was mentioned in a previous chapter, the best humanoid robots ever designed are HRP-2 constructed by Kawada Industries [Kaneko 2002] and the ASIMO constructed by HONDA [Sakagami 2002]. Let us make a small study (as much as limited open sources of information allow) of these robots.

ASIMO

It should be mentioned that there is a lack of information concerning the architecture of this robotics platform because it is part of commercial classified information and is protected by the manufacturer for competitive activity in the field of humanoid robotics.

First of all, the robot has an installed navigation system to move in structured environments and an audio/video system for human-robot interaction. Its approximate hardware architecture is presented in the figure 3.16.

The hardware architecture of the robot is provided by two PCs. The intelligence and the human - robot interaction is provided by PC2 with a frame grabber module allowing capturing video. PC1 provides motion planning and control functions for actuators. The same PC processes sound information using DSP modules and external communications. All modules inside the PC are connected by the PCI bus in order to transfer high velocity data. As the video processing system is separated from the motion control and planning system, it operates at a very high speed and doesn't occupy any general control resources and concentrates on the motion and stabilization control only.

The Ethernet is used for communication between different processes with different velocities and sample times. For example, image processing is a slow process which uses a lot of computational resources but audio processing or speech generation is a fast process. To allow the autonomous robot to communicate with external server, wireless communications are implemented. The communication server maintains sockets gaits for data exchange between vision and motion planning.

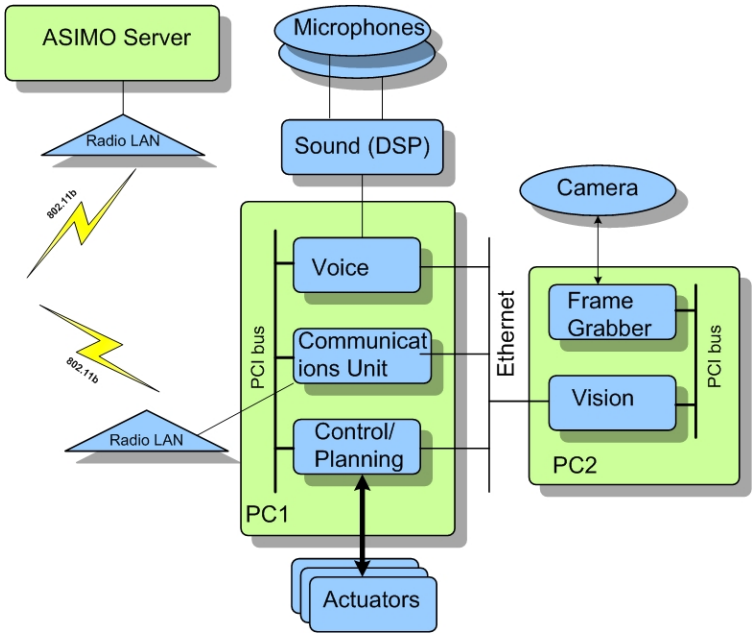


Fig. 3.16: Control architecture of ASIMO humanoid robot

As it can be observed from the previous figure the control hardware architecture is very generic and doesn't include detailed descriptions of how the bottom level, i.e. the locomotion system, was developed. Although the detailed hardware system of ASIMO was never presented in free sources of information we can suppose that it is based on the ideas prescribed in the HRP-2 robot because initially, the technology of bipedal locomotion was acquired by HONDA from providers of this system. More detailed hardware structure of HRP-2 is available and will be presented later.

The software system consists of motion planning and motion control modules. Also, it has video processing and speech processing modules. The motion planning module is controlled by events without any central control module. It should provide fast obstacle free planning using video information about the environment. Thus, in this robot the distributed control architecture is mainly based on implemented control agents. The motion control module controls walking stability and implements fast sensorial data processing.

The external server manages general maps for navigation in well structured environments like offices, museums or hospitals. This system sends upper level commands to the robot and allows the selection of tasks for more specialised activities such as speech recognition and dialogue with a user. It also provides the data base with images of different persons used for human identification and interaction.

ASIMO currently uses different operational systems and a messaging system for internal communication between modules and events.

HRP-2

The HRP-2 is the robotic platform for the Humanoid Robotics Project headed by the Manufacturing Science and Technology Center (MSTC) of Japan. The entire robotics system was designed and integrated by Kawada Industries, Inc. together with the Humanoid Research

Group of National Institute of Advanced Industrial Science and Technology (AIST). HRP-2's height is 154 cm and mass is 58 kg including batteries. It has 30 degrees of freedom (DOF) including two DOF for its hip.

The hardware architecture inside the robot uses control boards specifically designed for the robot. The communication bus was supposed as a PCI bus. The main problem was the availability of only 4 slots in a PCI bus for connecting 4 devices. Because of 30 DOF, the robot has DC 30 motors and 30 encoders. Moreover it has 3 gyro sensors and accelerometers and 4 force/torque sensors. All these devices should be controlled by 4 PCI control boards. As there were no conventional electronics components for humanoid robots, the manufacturer developed its own HRP-2 Interface Boards for motion control and a Quad F/T board for sensorial data processing.

Also, in order to provide the humanoid robot with actuators it is necessary to use some kind of servo controllers. As the number of active DOF is very high, there was a problem with the volume of weight of necessary electronic components. The manufacturer developed its own compact servo controllers with 15% less volume and 33% less weight compared with most compact devices available on the market. Each servo module controls 2 joints therefore there are 15 modules needed for effective control of the HRP-2 robot.

The use of the boards outlined above, allows the robot to implement the control scheme presented in the figure 3.17.

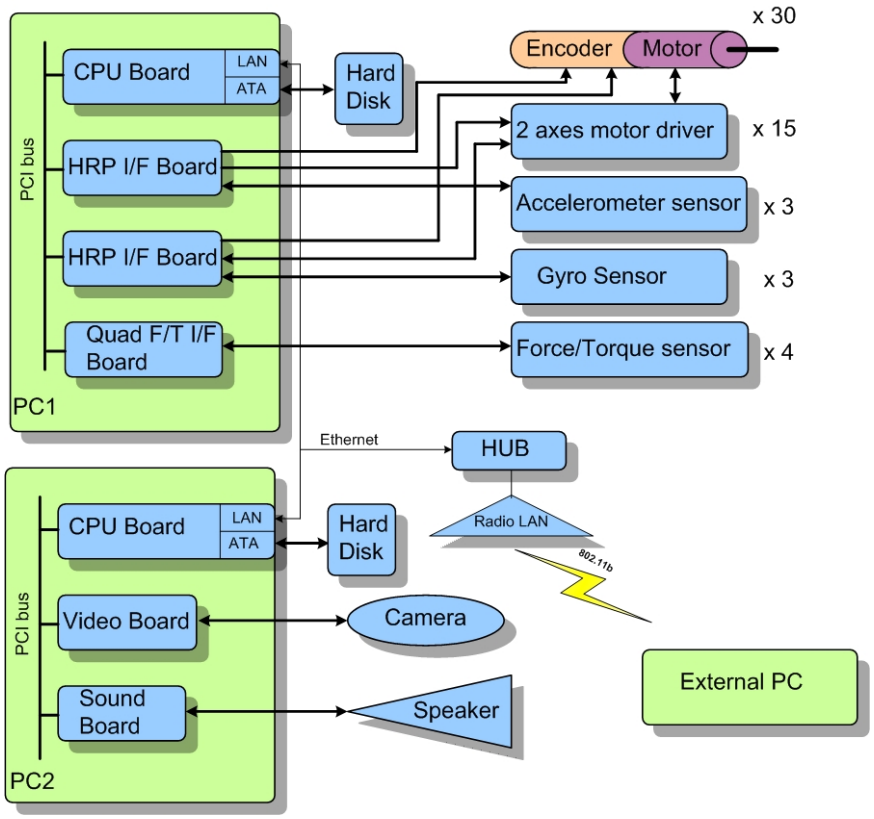


Fig. 3.17: Hardware control architecture of HRP-2 humanoid robot

This hardware system consists of two PCs. One of them is concerned with motion control and sensorial data processing functions and the other one is concerned with video and sound processing. The core of each system is a Pentium III de 1 GHz processor.

The network diagram of the network inside of the HRP-2 is shown below.

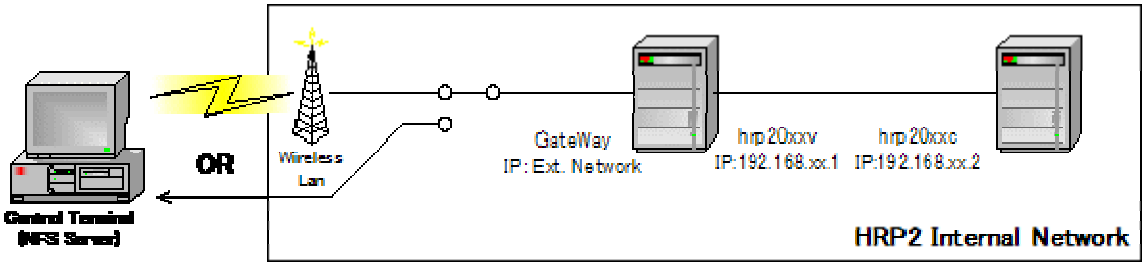


Fig 3.18: Robot Internal Network (from OpenHRP Manual)

The core Control System runs inside the computer with the hostname hrp20XXc. This host has the ability to communicate with sensors and joint motors through the HRP-2 Interface Board.

Attached to the computer with the hostname hrp20XXv are the cameras and speaker. This will be used mostly for vision processing and speech generation. On standard versions of the HRP-2 it also acts as the gateway for the network inside the HRP-2.

The HRP-2 is usually given commands by another PC which can be called the Console PC. The Auditor interface which will be explained in the following chapter runs on the Console PC, and monitors the state of the robot and sends it command scripts.

The software architecture for the HRP-2 robot is designed to be able to work with the ART-Linux operating system. The main advantage of the HRP-2 humanoid robot is the availability of the software platform OpenHRP which consists of a dynamics simulator, view (camera) simulator, motion controllers and motion planners. The OpenHRP is designed for using with real HRP-2 humanoid robots but it also can be used for dynamical simulations without humanoid robot functioning, simulating not only the dynamics of the robot but also the sensorial output. The users can develop their own software for OpenHRP as well as replace its building blocks with their own ones. In this work the standard Stabilizer plug-in for HRP-2 humanoid robots was replaced by the My_Stabilizer plug-in which contains the stabilization algorithm presented and discussed above. A more profound study of the software system the HRP-2 humanoid robot will be provided in the following chapters.

All features presented above make the OpenHRP system one of the best platforms for simulation and development of new motion control algorithms. Therefore, one of the basic parts of the control system designed and developed in this thesis will be tested with this platform (Chapter 7).

To conclude, it should be mentioned that the presented hardware and software systems provide the ASIMO and HRP-2 robot with practically the best walking skills and abilities for human-robot interaction. The next section will provide a discussion on the presented case studies and set out guidelines for the proposal of a new open control architecture for humanoid robots.

3.7.4 Discussion

Before starting the design of a generalized motion control architecture for humanoid robots let us pose some special requirements and specifications. In order to specify basic requirements for the motion control architecture of a humanoid robot it is necessary to discuss and compare basic characteristics of motion control architectures for the basic types of robots presented above. Based on the analysis and study of the different robots described in previous sections it is useful to create a table presenting basic characteristics that need to be taken into account for motion architecture design.

Robot Type	GDL	Joint Control	Synchronized multi-axis operations	Motion control autonomy	Additional motion control
Polyarticulated	5-7	+	+	-	-
Mobile	1-2	+	-	+	-
Humanoid	>21	+	+	+	+

Table 3.1: Basic motion control characteristics of principal types of robots

Humanoid robots usually have appreciably more degrees of freedom than mobile and polyarticulated robots. While joint control is usually implemented in every type of robot, the large number of DOFs leads to the need for synchronized multi-axis operations (the case in humanoid and polyarticulated robot). On the other hand, mobile and humanoid robots usually produce displacements in different environments. This leads to the need for autonomous motion control architecture, physically located inside the robot's body.

Finally, in the case of humanoid robots, the implementation of only joint control doesn't guarantee the stable locomotion. Evidently, even if all joints execute their motion patterns correctly, the robot can still overturn and fall down because of its dynamical instability (a more profound study of the bipedal locomotion of humanoids will be provided in the chapter 5). Therefore, humanoid robots need another (in addition to the joint control), more complex, control loop for maintaining its stable bipedal locomotion.

Overall, as can be observed from table 3.1, humanoid robots have, on the one hand, some requirements in common with mobile robots, and on the other hand they have other characteristics close to the polyarticulated manipulator. Moreover, they have some special functional features which require the implementation of special control methods and algorithms. Thus, the main designing consideration in open motion control architecture for humanoid robots will be the attempt to implement well proven solutions from other robotics fields, with adaptation for humanoid robot needs.

The design criteria for control architecture in general having been established in previous sections, let us now make things more concrete and clarify the concept of robot control, especially the control of humanoid robots. In general, a robot control system acts as a sort of artificial mind in order to direct achievement of a given goal. Therefore, a robot can be considered an artificial mind inside an artificial body. From the information processing point of view, a robot mind performs four basic functions: control, perception and decision making, including planning. Figure 3.19 illustrates a framework for the control architecture of a robotics

system. It can be divided into two principal parts - Physical state control and Mental development [Xie 2003].

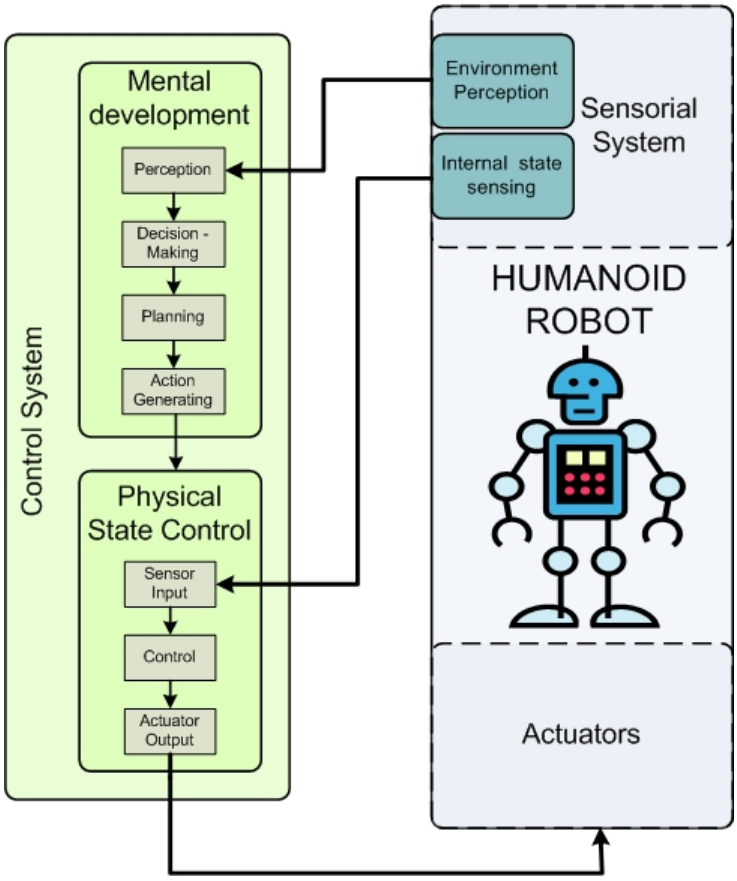


Fig. 3.19: Framework of a robot control architecture

Mental development function is the control activity which relates to the external world, with its internal representation inside the robot mind. The basic mechanism driving mental activities is the well known perception-decision making-action planning chain.

Perception is the key to intelligent robot behaviour [Fitzpatrick 2003]. It is the process of acquiring, interpreting, selecting, and organizing information from different sensors of a robot. Two basic types of robot perception can be distinguished. Active and interpersonal perception both act as sources of "special situations" which allow the robot to temporarily reach beyond its existing perceptual abilities, giving the opportunity for development to occur. Active perception refers to the use of motor action to simplify perception [Ballard 91], and has proven its worth many times in the history of robotics. It allows the robot to perceive things that it (initially) could not without the motor action. Interpersonal perception refers to mechanisms by which the robot's perceptual abilities can be influenced by those around it. Physically the perception is implemented by sensors and by dedicated processing of the data they produce.

Decision making can be considered a cognitive process leading to the selection of one type of action from those available. A final choice is the goal of every decision making process and can

be an action or an opinion. Decision making begins when we need to do something but we do not know what. Therefore, decision making is a reasoning process which can be rational or irrational, and can be based on explicit assumptions or tacit assumptions [Lu 2007]. Clearly, this decision making is an indispensable part of any autonomous robotics system.

From the *action planning* point of view, there are different kinds of architectures, taking into account efficiency, response time and the availability of environmental information. There is, for example, a purely strategic planning using precise environment model. This kind of architecture supposes that every situation which will face a robot is known and can be predicted a priori. On the other hand, there are purely reactive architectures, which implement a control strategy based on the action-reaction interaction. These systems contain a collection of reactive rules to be implemented in accordance with information perceived by the sensors. Behaviours based architecture [Arkin, 98] is an evolution of the reactive based approach. Behaviours describe the form of complex reaction on the changing of a sensor's state. Normally, there is a collection of parallel behaviours executing actions in a concurrent way. Of these various kinds of architectures it is necessary to mention "subsumption". This architecture was proposed by Brooks [Brooks 86] and is based on the idea of fragmentation of the control problem by using various behaviours to execute tasks. In this model, architecture consists of layers of behaviours, each one representing a more advanced level than the lower layer, which it acts over. Each behaviour is represented as a finite state machine which has a sensor's signals as its input and produces output signals for actuators. Subsumption means that the upper layer behaviour can subsume the functionality of the lower layer.

Apart from the mental development, there is another type of control. By the physical functions (or abilities), we usually mean all motions (actions) produced by a robot which are carried out by the sensory-motor system, controlled by a closed-feedback loop. This control can be considered a physical development function of robot control activity. It allows for the production of a motion, desired by the mental development part of the robot's control system.

The key concern of the present Ph. D. thesis work is the development of the state (motion) control architecture (taking into account some aspects of the mental control architecture) for the humanoid robot. In the following section, the concept of the motion control architecture for a humanoid robot will be explored in detail.

3.8 Motion control architecture of a humanoid robot

Basing on the previously described design principles (section 3.4) and discussion (section 3.7.4) let us consider the basic functional specifications of motion control architecture for a humanoid robot, which is the central theme of this Ph.D. thesis.

3.8.1 Conceptual level

On a conceptual design level as was mentioned above, we need to establish the means by which a motion control system executes a determined task. It is evident that a robot should execute its motion in a totally automatic way, allowing the human master restricted possibilities to monitor its motion state. In this context, the most appropriate concept for realizing the motion control of a humanoid seems to be an autonomous control (figure 3.20).

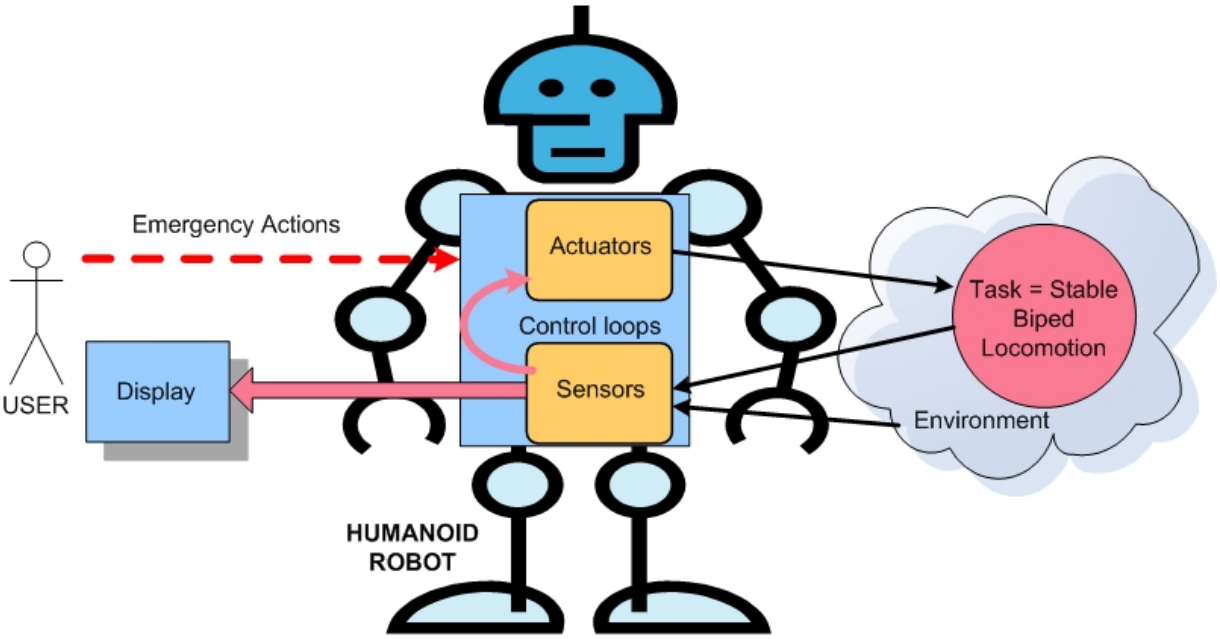


Fig. 3.20: Autonomous motion control of a humanoid robot

The main task of a motion control system is to provide stable biped locomotion for a humanoid in its environment. Actuators of the robot perform previously planned motions to achieve a synchronized multi-axis biped locomotion. Sensors mounted into the robot's body provide information about the environment and the actual state of the robot. A special algorithm closes the control loop. In this control scheme an operator can only specify some control parameters and observe the actual state of the robot but is unable to intervene in the control process because of its very high speed and complexity. Human beings are not able to operate at such low levels of control as the motion control of a humanoid requires. In the case of a critical emergency, the operator has the possibility of intervening (for example emergency stop of the robot) but only by interrupting all control operations of the robot.

3.8.2 Functional level

The functional design level usually consists of the development of a hierarchical structure of the control system. This hierarchical structure establishes all possible functional control levels of a humanoid robot. Motion execution and its control is only one part of the global task of a humanoid. In order to determine the significance of motion control, let us recall the figure from the previous section (figure 3.21).

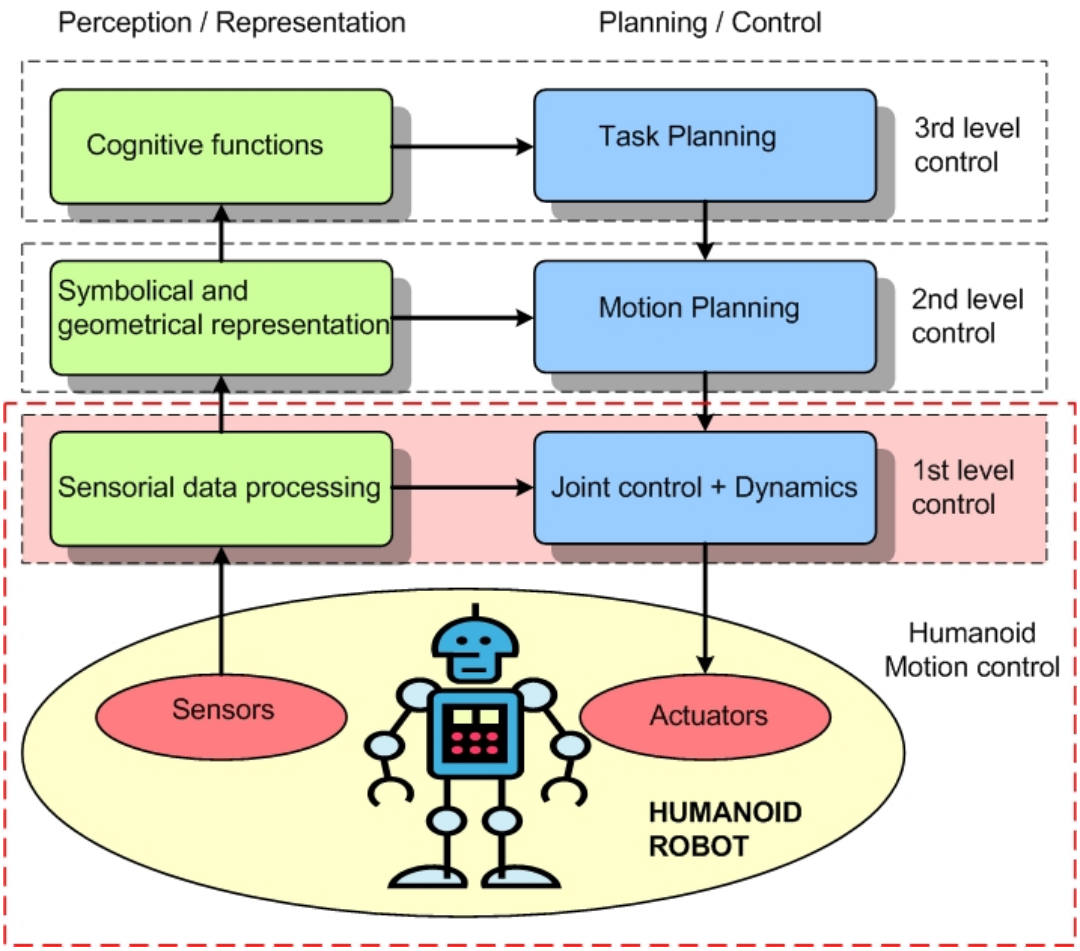


Fig. 3.21: Functional hierarchy of a control system

In this context, the motion control of a humanoid robot is the first functional level of a robot global control system. It is responsible for sensorial information acquisition, as well as control of joint trajectories and the entire dynamics of the robot's motion. Furthermore, this control level interacts directly with the mechanical and electronic systems of humanoids and includes control algorithms, as well as hardware devices and software components (these will be considered later when discussing the physical design of a motion control system).

In terms of control algorithms, the essential difference between humanoids and other kinds of robots is that the movement of the humanoid robot has to be human-like, using legged locomotion, and above all, a biped gait. The general motion control architecture comprising all necessary control algorithms for the motion of a humanoid robot is presented in figure 3.22.

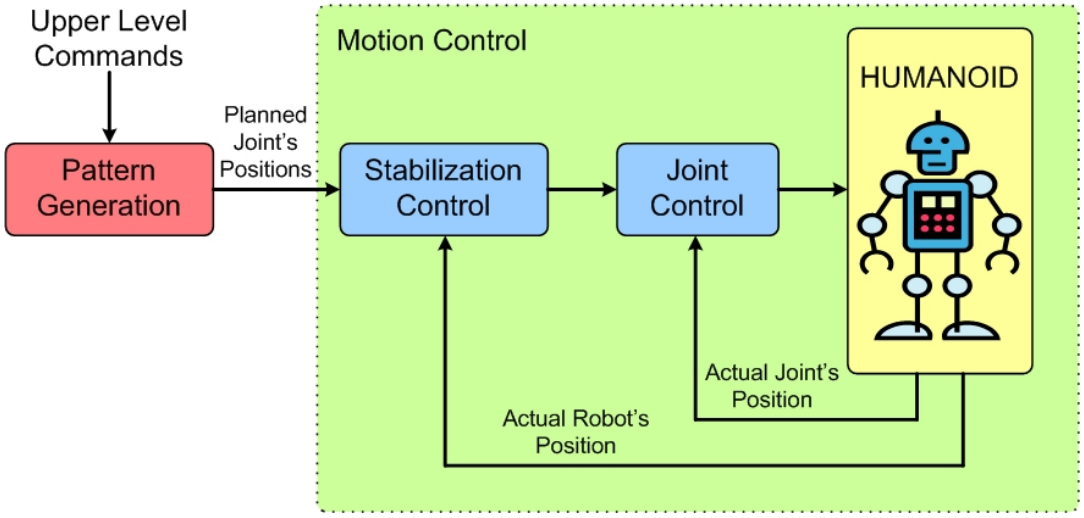


Fig. 3.22: General motion control architecture of a humanoid robot

All the motions of humanoid robots are usually planned by a motion pattern generator. Motion planning is activated by an upper level command from a human master or the upper level intelligence of a proper humanoid (mental development functions). Stable motion patterns, with reference to the joint positions, are sent to the robot. An essential element of a control system is a joint control mechanism allowing the correction of errors, positioned in every joint of the robot. This element of a motion control system is common to every type of robot. In other words, if we have a motion we always need to use a feedback to control it.

A humanoid robot differs from other types of robot because even if each joint follows a correct and well controlled motion pattern, this doesn't guarantee a stable biped locomotion. This is because while actuators perform angular motions, whole robot performs a bipedal translational motion with very complex dynamics of a mechanical structure. Thus, the implementation of another control element is needed. Stabilization control is a mechanism allowing a humanoid robot to make a stable biped motion.

Both of the control algorithms presented in figure 3.22 (joint and stabilization controls) are critical for humanoid robot motion and should be taken into account when designing an open motion control architecture. The following chapters, 4 and 5 will provide a detailed study and design of these control algorithms.

3.8.3 Physical level

On a physical level, humanoid robot control architecture may embody several different architectural styles and structures. The computation and communication basis reflects a particular style while the architectural structure shows how a system is divided into subsystems and how these subsystems interact between each other [Atta-Konadu 2005]. A well conceived architecture can have many advantages in the specification, execution, and validation of a robot control system.

As was mentioned in previous section 3.4.3, there are two basic considerations for physical design of control system architecture. They are the level of modularity and the degree of

distribution/centralization of functions and hardware/software components. In the case of the control architecture, dedicated only for control of the motion of a humanoid robot, the following considerations can be made. The architecture should provide:

- Continuous monitoring of a robot's state
- Real-time data acquisition and processing
- The execution of different simultaneous tasks such as, for example, joint and stabilization controls
- A high-speed safe data exchange between its components
- Simple redesign and addition of new functionalities to the architecture
- Easy connection with upper level controls (motion planning, decision making, etc.)

For systems with few degrees-of-freedom, centralized control performs very well and easily compensates for coupling between controlled axes. Its performance deteriorates as the number of axes increases, since control algorithms become more complex. Furthermore, its capacity to reconfigure is very poor because any change in the number of controlled DOFs or control algorithms could lead to a redesign of its entire structure. In contrast, a distributed architecture has its own controller for each task, thus allowing for lower communication and computation delays, and even for a large number of controlled DOFs. Designers of intelligent architectures generally prefer hierarchical architectures. In these architectures each layer of a hierarchy is used to simplify and abstract the problem for the layer above. It allows the higher layers to be more isolated from the hardware and thus, simplifies the design of the system. Also, it provides for better portability of the architecture to different hardware.

Thus, the most appropriate architecture for motion control of humanoid robots in both, software and hardware levels seems to be a modular distributed hierarchical (one master) architecture, with a main module responsible for task assignation and priority management (Figure 3.23).

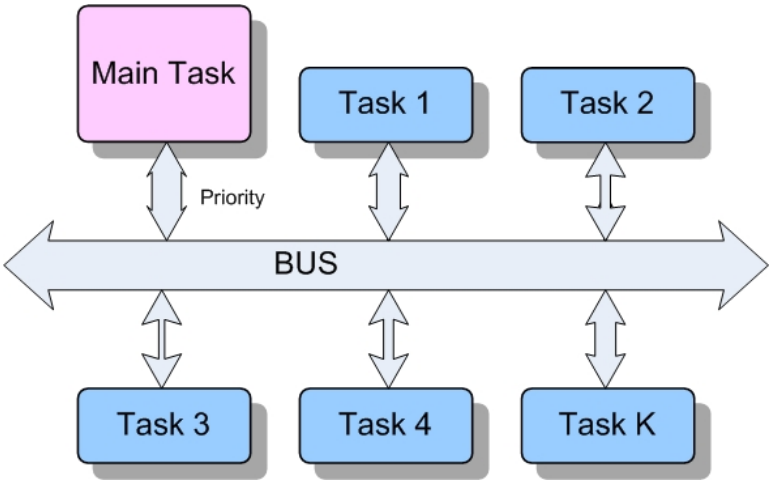


Fig. 3.23: Distributed hierarchical control architecture for humanoid robots

This architecture can easily fulfil all the conditions posed above and has the benefit of the distribution of its principal components, which allows simple scalability of the system. The main module provides the synchronization and the interaction with the upper levels of control.

The major benefits of modularity are a low effort in future redesign, and bigger flexibility of the architecture depending on the requirements of each application. This makes the control architecture very versatile and adaptable. Modularity reduces redesign efforts because each new module is not started from scratch, and thus, also reduces costs when the requirements of the application changes.

A more detailed design of the open motion control architecture for humanoid robots will be provided in chapter 6.

3.9 Conclusions

This chapter provides a complex study of control architecture design. Robot control architecture is a set of control algorithms which makes use of elements of both software and hardware and is used to enable the design of such a composite system. As motion is a key concept in the designing of a robot, the motion control architecture can be considered as its essential element.

Also, in this chapter, basic aspects which have an influence on the control architecture such as efficiency, programmability, adaptability, portability and the level of autonomy and evolution capacity, have been considered and different design levels of a control architecture discussed.

The main goal of every motion control system is to provide correct motion execution. Basic motion control is a servo joint control algorithm. This chapter provides a general review of principal joint motion control methods. In following chapter 4, based on this study, a joint motion control for humanoid robots will be designed.

A case study provided in this chapter compares principal types of contemporary robots (polyarticulated, mobile and humanoids) and presents the control systems of two of the most advanced contemporary humanoids – ASIMO and HRP-2. Based on this knowledge, a proposal for generalized open motion control architecture has been made. This autonomous motion control architecture is based on the modular distributed hierarchical structure. The benefit of this architecture is the possibility of easy and effective control of a lot of degrees of freedom, which is essential for humanoid robots as they usually have a multitude of DOFs. Moreover, this architecture is easy extensible and allows adding other types of control such as walking stabilization algorithms. This structure makes the architecture very versatile and adaptable.

Further design of motion control architecture for humanoid robots in this thesis work follows the chart presented in figure 3.24. The whole process is divided into 3 principal stages: concept, design and implementation.

This chapter provides a conceptual study of motion for humanoid robots. It proves that motion for a humanoid robot means biped locomotion. Moreover, this chapter poses the basic requirements for the design of motion control architecture.

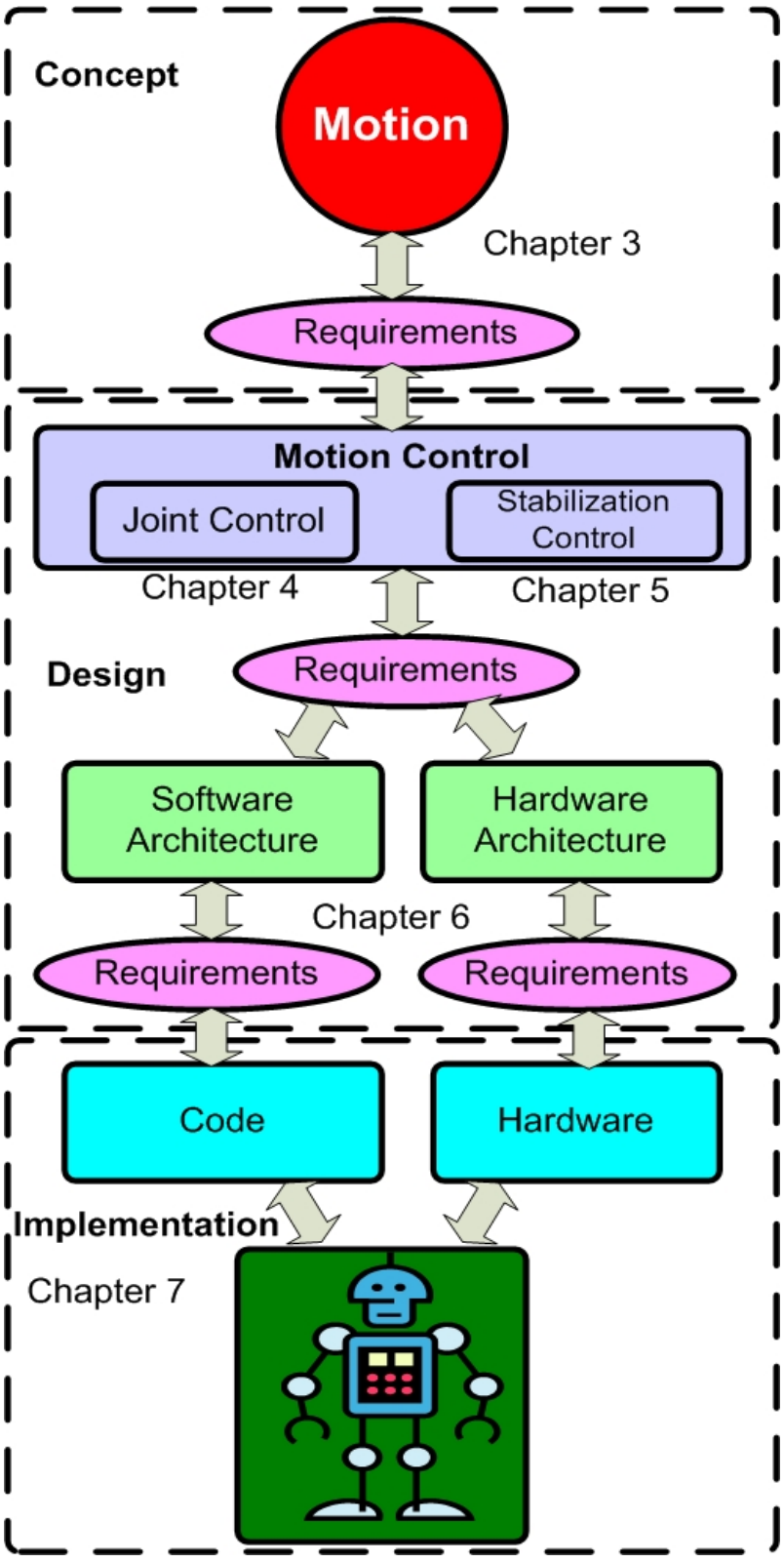


Fig. 3.24: Framework of the Thesis

Subsequent chapters will provide the design of architecture. As humanoid robots have a multi joint mechanical structure, this leads to different problems which should be solved by a motion control system. First of all is the joint control problem for a multi-axis system. The motion of every joint changes the dynamics of a robot and therefore, has an influence on other joints. Chapter 4 will discus joint control problems for humanoid robots. Also, the implementation of only servo control for some types of robots (especially for walking systems) is not sufficient. These robots need the implementation of another, upper, control loop which will provide their stability of motion. Chapter 5 will propose the design of a stabilization system for humanoids, which will provide them with a stable bipedal locomotion. And finally, there remains a synchronization problem. Every joint is working with its own servo controller but should participate in synchronized multi axis operations. Chapter 6 will provide the detailed design of hardware and software architectures as well as a communication infrastructure for such systems.

The implementation of the designed motion control architecture, as well as some walking experiments, will be provided in the Chapter 7.

Chapter 4

Joint control of a humanoid robot

CHAPTER 4

Joint control of a humanoid robot

4.1 Introduction

As stated in the previous chapter, joint control is usually a kernel of the global control system of a humanoid robot that also includes stabilization control - essential condition for achieving stable bipedal walking.

There are a lot of works related to the joint control of different kinds of manipulators [Wu 85], [Grabbe 92], [Lewis 2003]. A humanoid robot is a more complex system from a control point of view. It usually has a lot of degrees of freedom (DOF) (for human-like bipedal locomotion it is necessary to control at least 12 joints in two legs) and suffers more disturbances because of high vibration and external forces when the robot is walking. These forces usually appear due to the effect of gravity on the robot's mass. Also centrifugal and Coriolis forces may be included due to the robot's complex motion. In addition, the load of the joints changes during the trajectory because the robot lifts and places its legs during periodical biped movement.

Unfortunately, nowadays there are few works addressing the problem of joint control for humanoids [Kee 2004], [Yang 2005]. This chapter considers a joint controller design for joint control and ascertains the need for adaptive control algorithms to be implemented with a humanoid robot.

Robot dynamics describes how a robot moves in response to forces provided by its actuators. To simplify the task, assume that actuators have depreciatingly small dynamics in comparison with the inherent mechanical characteristics of the entire robot. In order to maintain the real trajectory of each joint similar to the ideal (motion pattern), the control has to be implemented. As the model is nonlinear, the design of the control system extremely complicated. In practice, some simplifying assumptions can be made. In most humanoid robots a distributed motion control system is implemented. In this kind of control system, each joint is controlled individually without taking into account the influence of others. This type of control clearly requires a model that considers each joint as an independent link associated with a motor. On the other hand, it is possible to implement this type of control only when an indirect transmission system is used (gear system). If the gear ratio is large, the motion of each joint could be considered independent from others.

Using an indirect transmission, the link dynamics is separated with respect to a motor. Each motor and gear system has its own dynamics that should be combined with the dynamics of the associated link. Taking into account the dynamical parameters of links and motors and friction effects associated with the transmission of motion, the robot from the control point of view can

be considered a black box with unknown dynamical parameters. Therefore, the first part of this work is dedicated to the identification of the dynamical parameters of a humanoid robot's joint.

After a model is designed, it can be used to design a controller. The objective of a control is the fast and precise positioning of a joint. The basic specifications of a controller are transient response and accuracy.

On the one hand, the different postures of a humanoid robot change motor dynamics. Variations of system's parameters and the magnitude of perturbations could disturb control performance. In order to maintain the desired system characteristics, adaptive algorithms to improve the control of each joint have to be considered. The goal is to achieve an adequate system response for the desired humanoid robot movements. Moreover, supplementary studies should be carried out in order to determine the need of adaptive joint control in humanoid robot bipedal walking.

All identification and control design presented in this chapter was created using the mechanical structure and motors of the Rh-1 humanoid platform, which will be described in further detail in chapter 7.

4.2 Joint Modelling

A humanoid robot requires precise velocity and position motor performance for joint movement. Each joint of a humanoid robot should precisely execute its trajectories. Because of disturbances mentioned above, this is not a simple task. Most humanoid robots have servomotor systems that use a DC motor with a permanent magnet. Figure 4.1 shows joint configuration of the humanoid robot Rh-1 used for experiments. This kinematical configuration, especially the bottom section containing legs, is typical (more or less) of the majority of contemporary robots. Therefore, the joint controller design provided in this chapter is applicable to any humanoid robot.

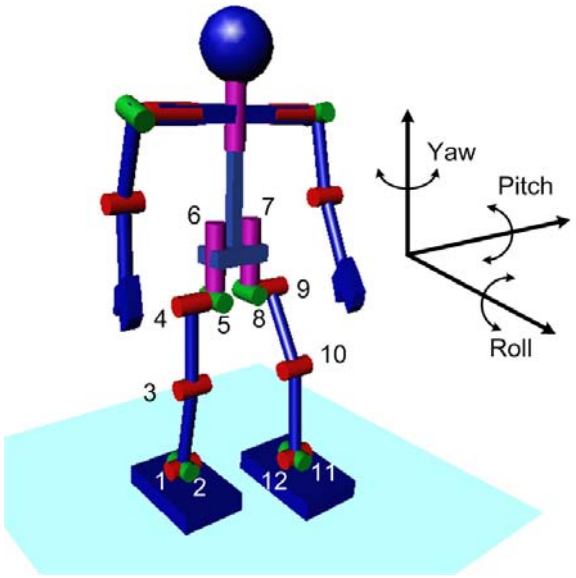


Fig. 4.1: Joints' configuration of a humanoid robot

As is seen in figure 4.1, the humanoid robot usually has 12 DOFs in its bottom section that are responsible for biped locomotion. There are 3 DOFs for the hip joint (roll, pitch and yaw) 1 DOF for the knee joint (pitch) and 2 DOFs for the ankle joint (roll and pitch). Each joint is compounded by the motor and reduction system. Reduction is usually compounded by the harmonic drive and belt transmission in order to transmit motion from the motor axis to the real axis of a joint (due to space restrictions it usually is not possible to place the motor directly in the joint axis), as shown in figure 4.2.

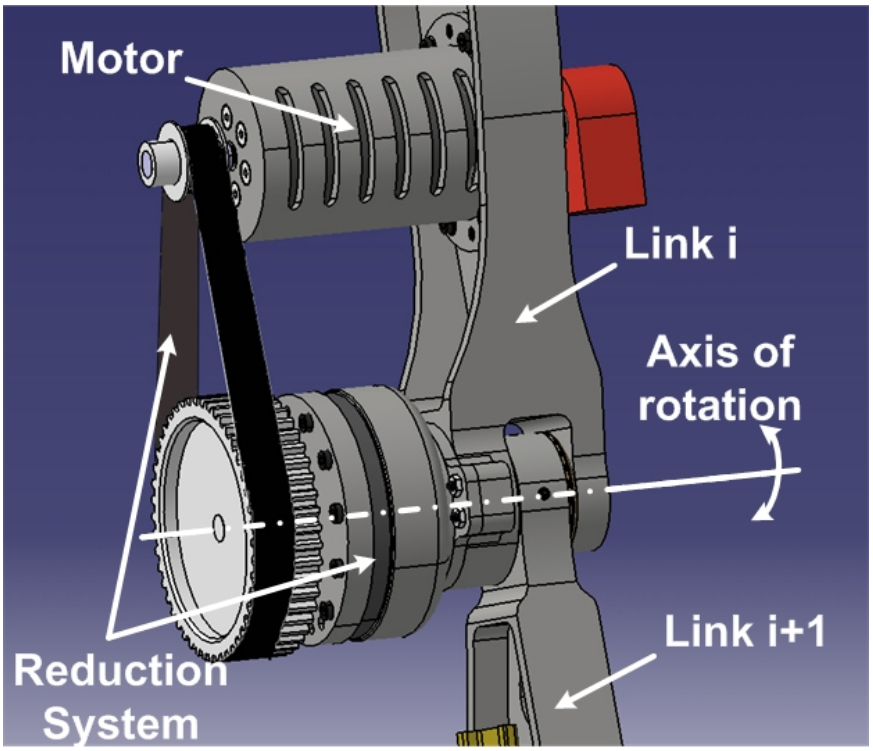


Fig. 4.2: Typical joint of a humanoid robot

The next section will study and propose the most adequate way of modelling humanoid robot joint control.

4.2.1. Complete dynamical model of a manipulator

It is necessary to introduce a complete dynamical model of a manipulator before starting the design of a simple joint model. It is important to study why it is possible to implement simplified model and detect possible restrictions posed by the segregating of joints' dynamics.

The following equation determines the complete dynamical model of a manipulator, including forces and torques τ provided by actuators to follow joints' trajectory θ.

$$\tau = I(\theta)\cdot\theta''+F\cdot\theta'+G(\theta) \tag{4.1}$$

Where τ , G , and F are n element vectors and $I(\theta)$ is a $n \times n$ manipulator inertia matrix. τ is a vector of gravitational terms and F is a vector of friction. Vector F only depends on derivatives of the corresponding articulation and can, therefore, be separated. $I(\theta)$ and G have as their coefficients expressions with terms related to the concentrated masses and distances between them, i.e. it depends on the articulation variables.

On the other hand, the influence of a reduction coefficient (gear ratio) implemented in each humanoid robot joint should be taken into account. If there is a diagonal matrix K containing gear ratios $k_{ii} > 1$, then, the following equations can be applied:

$$\theta'_a = K \cdot \theta' \tag{4.2}$$

$$\theta''_a = K \cdot \theta'' \tag{4.3}$$

$$\tau_a = K^{-1} \cdot \tau \tag{4.4}$$

where τ_a determines resulting torque in the motor axis.

On the other hand, inertial matrix $M(\theta)$ can be determined as follows:

$$M(\theta) = M_1 + M_2(\theta) \tag{4.5}$$

Where M_1 is a diagonal matrix composed of constant elements of the diagonal matrix $M_2(\theta)$ and indicates the contribution of each link in the inertia supported by a corresponding motor.

Substituting (4.2),(4.3),(4.4) and (4.5) in the complete dynamical model of manipulator (4.1):

$$\tau_a = K^{-1} \cdot (M_1 + M_2) \cdot K^{-1} \cdot \ddot{\theta}_a + K^{-1} \cdot G + K^{-1} \cdot F \cdot K^{-1} \cdot \dot{\theta}_a \tag{4.6}$$

this can be written as follows:

$$\tau_a = K^{-1} \cdot M_1 \cdot K^{-1} \cdot \ddot{\theta}_a + F_a \cdot \dot{\theta}_a + \tau_p \tag{4.7}$$

Where

$$F_a = K^{-1} \cdot F \cdot K^{-1} \tag{4.8}$$

is a diagonal matrix of viscous friction constants in actuators and

$$\tau_p = K^{-1} \cdot M_2 \cdot K^{-1} \cdot \ddot{\theta}_a + K^{-1} \cdot G \tag{4.9}$$

can be considered perturbation torque – the consequence of the interaction between manipulator's articulations.

The equation (4.9) shows that if the reduction coefficient is high, i.e. motion velocity of each motor is appreciably higher than the motion of a link, then the elements of the matrix K^{-1} will be very small. In said case, the perturbation torque τ_p is small and can be depreciated. Therefore the motor torque in motor i will take effect only upon the motion of articulation i remaining defined by the second order differential equation (4.7) obtained above.

This result allows us to evaluate to what extent a humanoid robot can be considered as a series of dynamically independent (decoupled) links in which the movement of one does not affect the others. For example, humanoid robot Rh-1 which will be described in detail later, has a Harmonic drive reduction gear with a 160 reduction coefficient, and some articulations have a belt with a 4 reduction coefficient (in total 640). Experiments provided with this robot confirmed the possibility of using decoupled stand alone joint models.

On the other hand, the introduction of a very high reduction coefficient is not recommended because it increases viscous friction and introduces a backlash in the transmission system and nonlinearities.

4.2.2. Decoupled joint model

Different methods are used to formulate a dynamical model of an entire humanoid robot. The control philosophy, implemented in most contemporary humanoids, is a distributed joint control. In this architecture, each joint is controlled independently by a motion controller. Therefore, it is convenient to study an individual dynamical model for each joint controlled by a motor. This model should consider each articulation individually without taking into account the others. In each joint, the dynamics is a combination of motor dynamics and attached link. For example, for the motor 9 (Fig. 4.1), an attached link is the whole leg of the humanoid robot.

DC motors is usually implemented to drive a humanoid robot joints. DC motors used in humanoids are usually coreless. This means that the moment of motor inertia is very small.

Figure 4.3 shows the scheme of a loaded joint of a humanoid robot. The electrical equivalent of a DC motor's rotor is represented as a resistor R_a and inductance L_a connected serially with the induced voltage V_a. On the other hand, considering the mechanical characteristics of the system, the motor moves a load corresponding to a joined link, which may be compounded by several articulations. In figure 4.3, it is represented by a moment of inertia J, viscous friction D and load torque T_L.

Moreover, the link is joined to a motor by a reduction gear playing an important role in the model. On the one hand, it attenuates the dynamics of the link sensed by the motor and contributes to the control, but on the other, it produces nonlinearities and hysteresis effects. This could affect the system, but the model presented doesn't take it into account because of its small magnitude and models the reduction gear as a constant multiplying of torque and dividing velocity of a rotating axes.

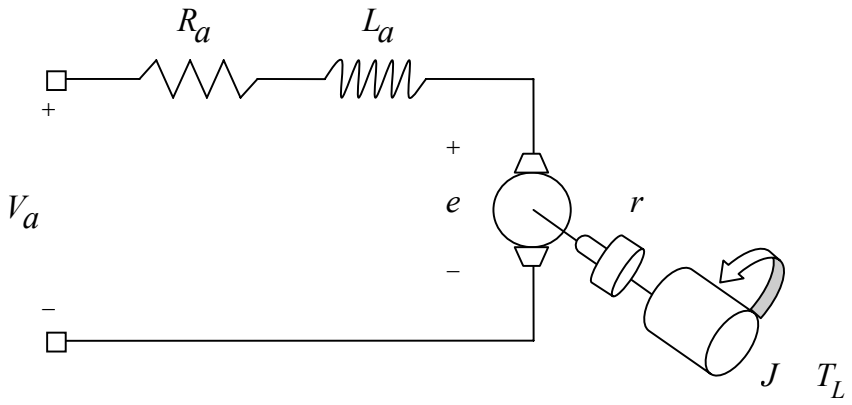


Fig. 4.3: Scheme of a loaded joint

Taking into account the electrical characteristics of the model and applying Kirchoff's 2nd law:

$$v_a = e + R_a \cdot i_a + L_a \cdot \frac{\partial i_a}{\partial t} \tag{4.10}$$

By a Faraday law, the electromotive force (emf) induced in an electric circuit is:

$$e = K_s * \Omega(t) \tag{4.11}$$

where Ks is a velocity constant determined by the density of a magnetic flow, $\Omega(t)$ - rotor velocity of a DC motor.

Taking into account the mechanical characteristics of the model, it is possible to construct an energy balance equation:

$$T_d = T_{W'} + T_W + T_L \tag{4.12}$$

where T_d is electromagnetic torque, $T_{W'}$ - inertial torque, T_W - damping torque, T_L - load torque.

Electromagnetic torque is proportional to the motor current:

$$T_d = Ks \cdot i_a \tag{4.13}$$

where Ks is a torque constant (same as velocity constant)

Inertial torque appears because of the motor's angular acceleration:

$$T_{W'} = J \cdot \frac{\partial \Omega(t)}{\partial t} \tag{4.14}$$

J represents the moment of inertia of a rotor and load (joined link) and is equivalent to the $I(\theta)$ discussed above. It depends on variables of articulation, but does not include inertial torques generated by acceleration in other articulations.

Damping torque is proportional to the rotor velocity and represents a dissipation torque:

$$T_W = D \cdot \Omega(t) \tag{4.15}$$

where D - represents viscous friction or viscous damping inherent in the system.

Substituting (4.13), (4.14) and (4.15) into (4.10) we can obtain the second differential equation of the model:

$$Ks \cdot i_a(t) = J \cdot \frac{\partial \Omega(t)}{\partial t} + D \cdot \Omega(t) + T_L \tag{4.16}$$

Equations (4.10) and (4.16) are lineal if their parameters are lineal. Therefore, the Laplace transform can be implemented.

$$V_a(s) = K_s \cdot \Omega(s) + I_a(s) \cdot (R_a + s \cdot L_a) \tag{4.17}$$

$$K_s \cdot I_a(s) = (D + s \cdot J) \cdot \Omega(s) + T_L \tag{4.18}$$

From (4.17) and (4.18) we can obtain a joint model:

$$\begin{aligned} \Omega(s) = & \frac{Ks^2}{(Ra + s \cdot La) \cdot (D + s \cdot J) + Ks^2} \cdot V_a(s) - \\ & - \frac{(Ra + s \cdot La)}{(Ra + s \cdot La) \cdot (D + s \cdot J) + Ks^2} \cdot T_L(s) \end{aligned} \tag{4.19}$$

Figure 4.4 shows a block diagram representation of the obtained dynamical model.

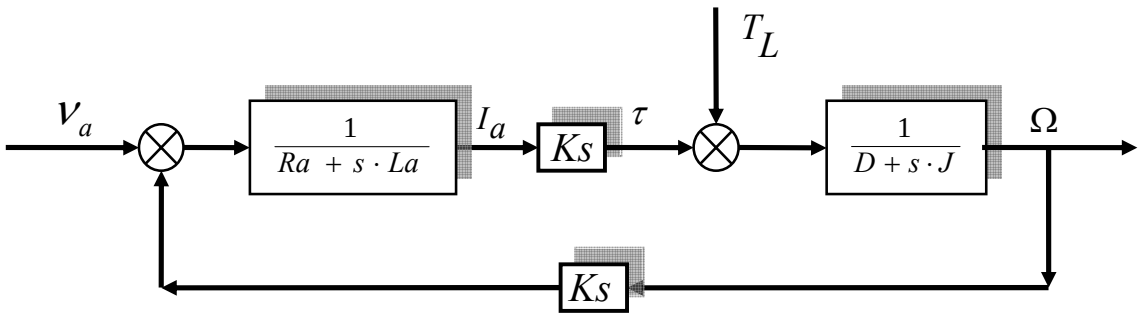


Fig. 4.4: Block diagram of a joint model with DC motor

A control variable for the system is usually a current, so the transfer function transforms into the next equation:

$$\Omega(s) = \frac{Ks}{(D + s \cdot J)} \cdot I_a(s) - \frac{1}{(D + s \cdot J)} \cdot T_L(s) \tag{4.20}$$

By this dynamical model, the system's velocity response is the sum of two components. The first depends on the rotor current I_a and the second on the load torque T_L. Load torque T_L and the moment of the inertia J depend on the mass distribution of the joined link or the posture of the humanoid robot. These two variables influence the model differently. The moment of inertia directly affects the dynamics of the system, while the load torque is uncontrolled system's input perturbing the relation (basically linear) between velocity and current of the motor. Implemented motors usually differ in nominal torques, but the obtained model is useful for each joint of a humanoid robot. In general, linear approximation is adequate in most cases, although it is difficult to predict to what extent. Experiments described in the following section solve this problem.

4.3 Identification design

The identification design consists of the estimation of a system's model based on observations of input-output data. In that way, a mathematical model describing dynamical behaviour of the system can be obtained.

Theoretically, it is preferable to identify a system under normal operational conditions, in a closed loop. In this case, we are facing the problem of a lack of the information about the system because the basic feedback goal is to make a system insensible to the changes. Identification methods applicable to the open-loop are generally not suitable for closed-loop identifications. The experiment carried out with the Rh-1 humanoid robot in a closed-loop did not provide good results. Finally, the identification was carried out in an open-loop.

Another issue is the type of input signal, i.e. its form and frequency. The preferable input signal is, of course, a pulse signal that excites whole range of frequencies equally, and the output is the system transfer function. As it is difficult to implement in practice, moreover, it can damage the motor, the input we used was a conventional step signal. This input allowed us to obtain simple yet precise models, especially in the case of a single variable process (as is the considered process).

In order to generate an input step signal, a command current with opposite signs and a different amplitude and frequencies was periodically sent to motors of the humanoid robot Rh-1. Figure 4.5 shows the typical signal used for the identification of a dynamical joint model.

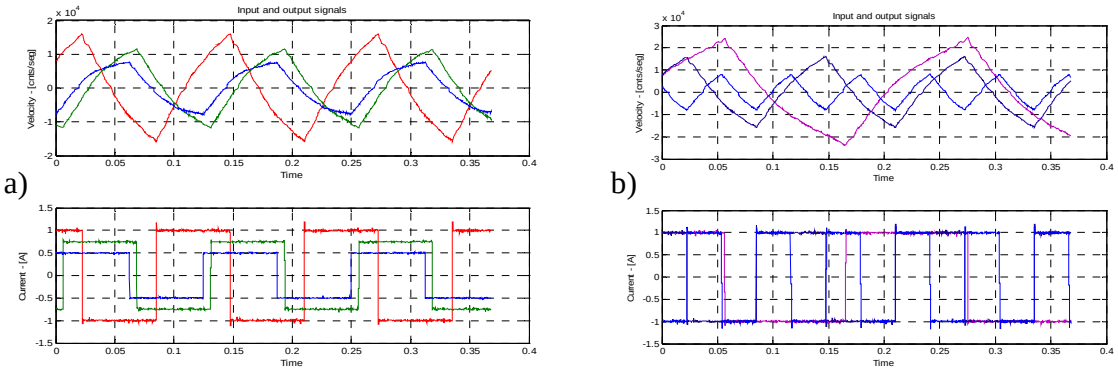


Fig. 4.5: Motor 10 velocity output for the step current input (sample time 360 μs) a) amplitude modifications b) frequency modifications

The initial study of the system was carried out with the Matlab System Identification Toolbox, a very powerful control design tool.

A general input-output linear model for a single-output system with input u and output y (Fig. 4.6) can be written as:

$$A(q)y(t) = \sum_{i=1}^{nu}\left[B_i(q)/F_i(q)\right]\cdot u_i(t-nk_i) + \left[C(q)/D(q)\right]\cdot e(t) \tag{4.21}$$

where y and u_i are the output and input sequences, respectively, and e is a white noise sequence with zero mean value. The polynomials A, B, C, D, F are defined in terms of the backward shift operator (z or q).

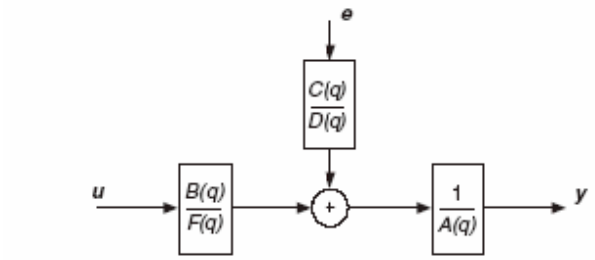


Fig. 4.6: General-Linear Model Structure

The general structure can be defined by giving the time delays nk and the orders of these polynomials. Most often the equation (4.21) is transformed into one of the following special cases (Fig. 4.7):

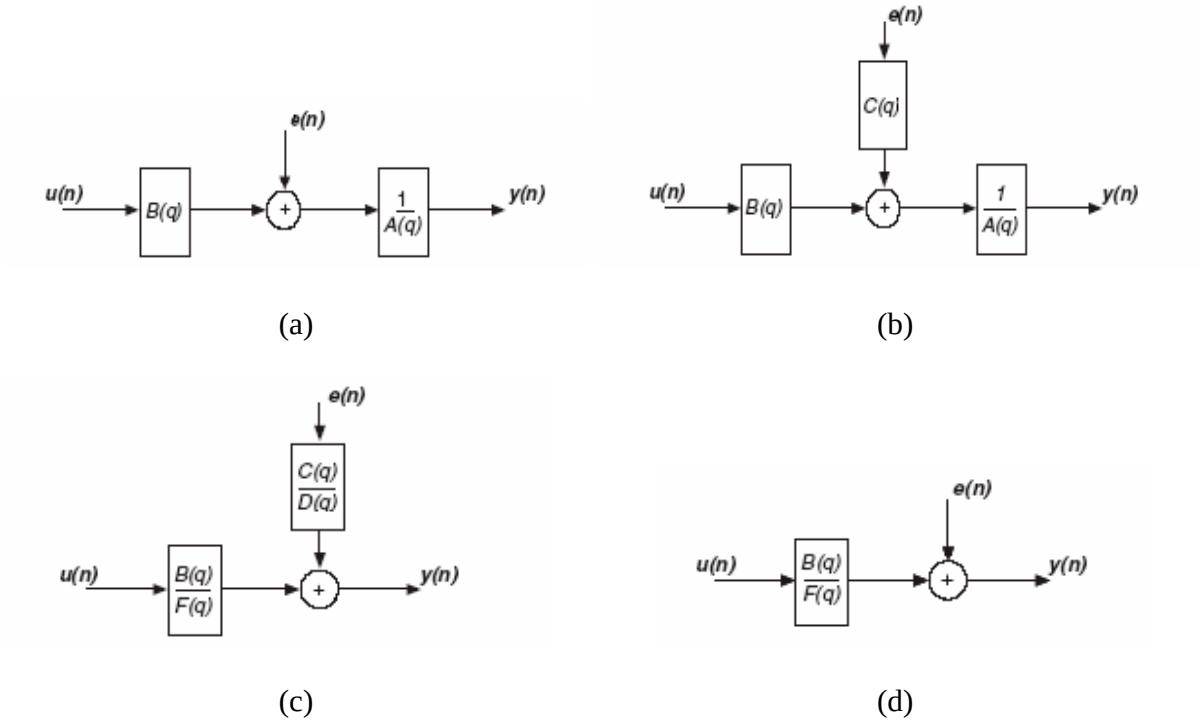


Fig. 4.7: Input-Output Model Structure a) ARX b) ARMAX c) OE d) Box-Jenkins

The ARX model, shown in Figure 4.7(a), is the simplest model incorporating the stimulus signal. The estimation of the ARX model is the most efficient of the polynomial estimation methods because it is the result of solving linear regression equations in analytic form. Moreover, the solution is unique. In other words, the solution always satisfies the global minimum of the loss function. The ARX model therefore is preferable, especially when the model order is high. The disadvantage of the ARX model is that disturbances are part of the system dynamics. The equation (4.22) describes the ARX model:

$$A(q)y(t) = B(q)u(t-nk) + e(\mathrm{t}) \tag{4.22}$$

Unlike the ARX model, the ARMAX model structure includes disturbance dynamics. ARMAX models (Fig. 4.7(b)) are useful when you have dominating disturbances that enter early in the process, such as at the input. The ARMAX model has more flexibility in the handling of disturbance modelling than the ARX model. The equation (4.23) describes the ARMAX model:

$$A(q)y(t) = B(q)u(t-nk) + C(q)e(\mathrm{t}) \tag{4.23}$$

The Output-error (OE) model structure (Fig. 4.7(c)) describes the system dynamics separately. No parameters are used for modelling the disturbance characteristics. The equation (4.24) describes the OE model:

$$y(t) = \left[B(q)/F(\mathrm{q})\right]\cdot u(t-nk) + e(\mathrm{t}) \tag{4.24}$$

The Box-Jenkins (BJ) structure (Fig. 4.7(d)) provides a complete model with disturbance properties modelled separately from system dynamics. The Box-Jenkins model is useful when you have disturbances that enter late in the process. For example, measurement noise on the output is a disturbance late in the process. The equation (4.25) describes the BJ model:

$$y(t) = [B(q)/F(\mathrm{q})]\cdot u(t-nk) + [C(q)/D(\mathrm{q})]\cdot e(\mathrm{t}) \qquad (4.25)$$

First, estimations of the delay and the order of the model using the simplest ARX structure were carried out. Then, different model structures (ARX, ARMAX, OE and BOX-JENKINS) were analyzed in order to determine the most suitable to be implemented in the Rh-1 joint identifications.

Figure 4.8 shows results obtained for the motor 10 applying different model delays and table 4.1 shows the best fitting.

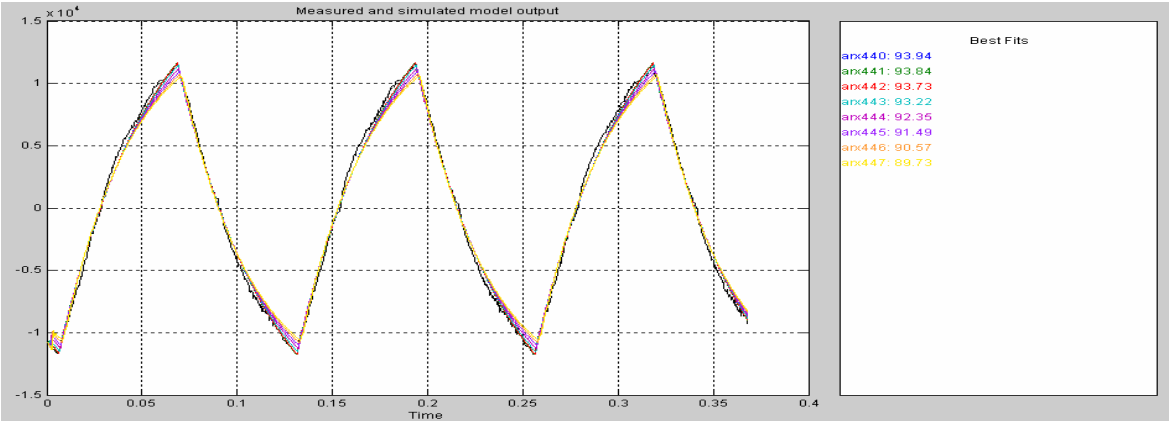


Fig. 4.8: Model delay analysis. Motor 10 (sample time 360 μs)

Model	Fitting
ARX440	93.94 %
ARX441	93.84 %
ARX442	93.73 %
ARX443	93.22 %
ARX444	92.35 %
ARX445	91.49 %
ARX446	90.57 %
ARX447	89.73 %

Table 4.1: Model delay analysis. Motor 10 (sample time 360 μs)

The results shown in the figure 4.8 clearly demonstrate that the model delay tends to the zero. This means that the input in moment k influences the output of the same moment k. As every real system should have a certain delay, the results obtained may be the consequence of a system delay smaller than the used sample time (360 μs).

The order of the model depends on the level of system differentiation with respect to the output signal. Theoretically, in depreciating system dynamics, the output relates to the input by the first order differential equation (4.16). Therefore, the system could also be the first order. In practice, the relationship is not so simple and other aspects should be taken into account, for example internal amplifying blocks implemented in the current controller built into the motors of the Rh-1 humanoid robot. The advantage of using black box models is the fact that we don't have to worry about elements increasing the order of the system. The objective is to find the simplest model able to adequately describe the system's response. For this reason, the major order of the model used in the experiment is 5.

Figure 4.9 shows the results obtained for motor 10, applying different orders of the model, and table 4.2 shows the best fitting.

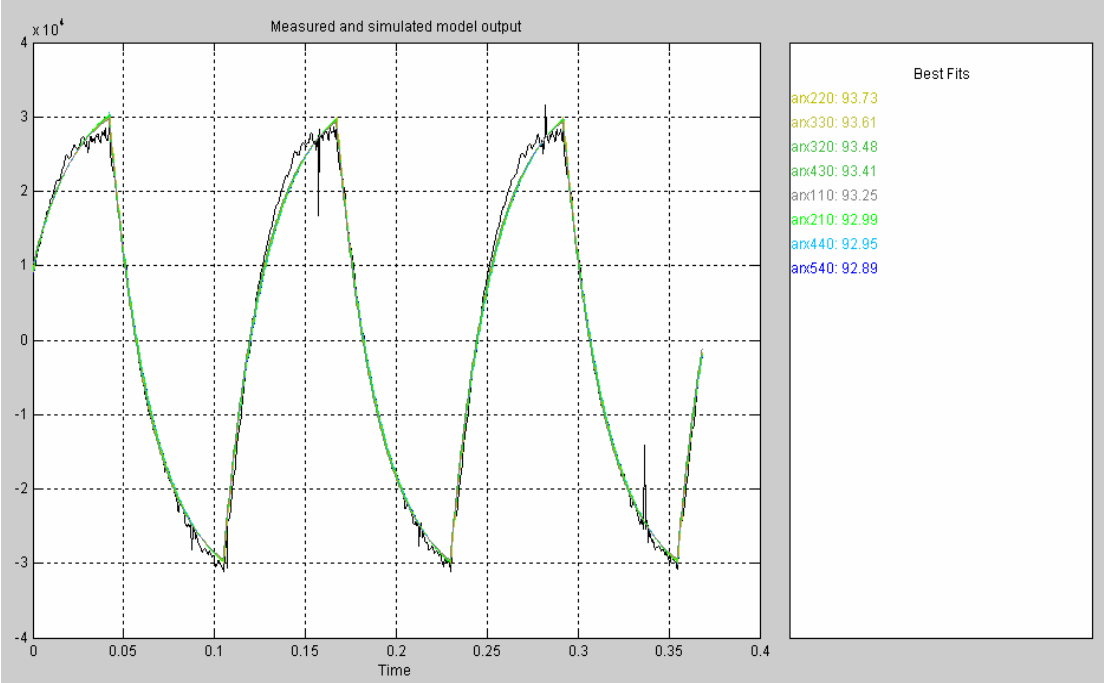


Fig. 4.9: Model order analysis. Motor 10 (sample time 360 μs)

Model	Fitting
ARX220	93.73 %
ARX330	93.61 %
ARX320	93.48 %
ARX430	93.41 %
ARX110	93.25 %
ARX210	92.99 %
ARX440	92.95 %
ARX540	92.89 %

Table 4.2: Model order analysis. Motor 10 (sample time 360 μs)

Figure 4.9 shows that the fitting is very good for every model order. In some joints, fitting of the first order model is almost the same as that of the higher order models. This shows that it is not necessary to use a complicated high order model. However, use of the first order model can be dangerous. Sometimes the signal can have more noise or the influence of the load torque and disturbances could be higher than realized in experiments. In these cases, first order models fail. We propose using a second order model (two parameters in denominator = two poles) since there is an insignificant difference between it and higher order models.

Figure 4.10 shows the ARX, ARMAX, OE and BOX-JENKINS model structures comparison, and table 4.3 shows the best fitting of these model structures.

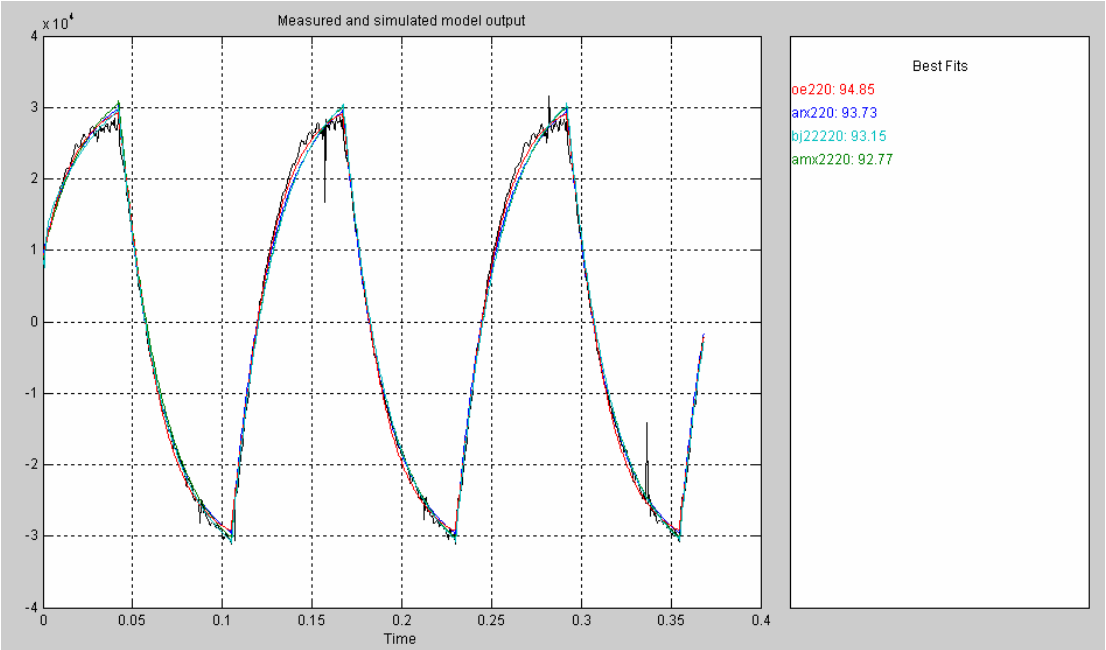


Fig. 4.10: Models analysis. Motor 10 (sample time 360 μs)

Model	Fitting
OE220	94.85 %
ARX220	93.73 %
BJ22220	93.15 %
AMX2220	92.77 %

Table 4.3: Model type analysis. Motor 10 (sample time 360 μs)

It is evident that the OE model shows the best results because it models disturbances more realistically. The second best is the ARX model, followed by the Box-Jenkins model and, lastly, the ARMAX model structure.

Note that the ARX model structure shows good results being the most simple model to estimate (does not require recursive estimation algorithm). Moreover, our experience shows that when basic properties of the system are obtained there is no reason to continue a fine tuning of

the model in order to gain a decimal of a percent of fitting. Therefore, the ARX structure can be chosen to model the joints of a humanoid robot.

One of the objectives of joint identification is the possibility of implementing all developed algorithms online (without using Matlab) with the hardware and software inside of the robot. For this purpose, an online identification software system was created.

The system acquires motion data from each motor of the humanoid robot and than uses the LS (Least Squares) algorithm for the ARX structure, with two parameters in numerator and denominator, and zero delay to estimate model parameters.

Equation (4.26) shows the mathematical representation of the estimated system for joint 10 (left knee) of the Rh-1 humanoid robot and figure 4.11 shows its root-locus:

$$ARX\ 220_{\ MOTOR\ 10} = \frac{129.8 - 119.3 \cdot z^{-1}}{1 - 1.945 \cdot z^{-1} + 0.945 \cdot z^{-2}} \tag{4.26}$$

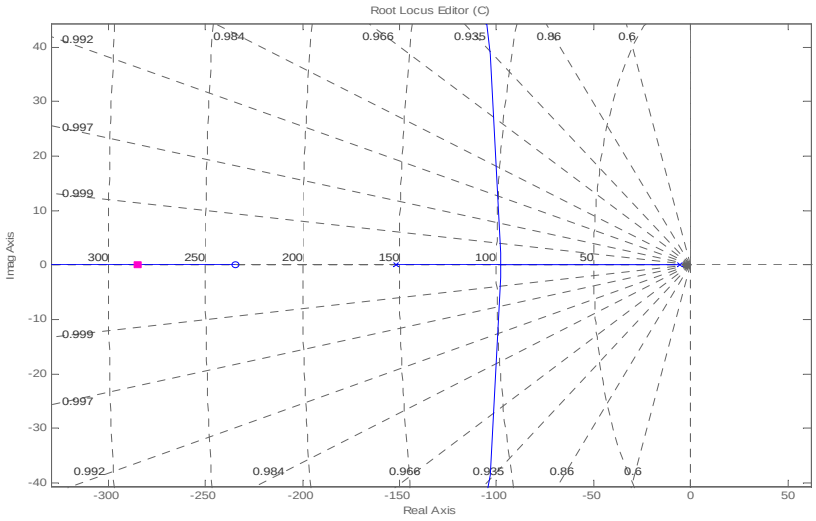


Fig. 4.11: Root-locus of the estimated ARX220 transfer function of the motor 10

4.4 Controller design

4.4.1. Theoretical Issues

The simplest strategy for joint control is the independent control with SISO (Single-Input, Single-Output) dynamical model presented above. Classical control theory provides very good results for most systems. Conventional methods, for example root-locus and frequency response, are applicable only in these cases. They are not useful for optimal and adaptable control designs because these are basically used in non-lineal systems.

The basic property of any control system is stability. Besides absolute stability, a system should have a reasonable relative stability, i.e. the system has to manifest a reasonable damping

and velocity response. It also has to decrease errors to zero or a small acceptable value. The condition of the relative stability and the precision of the system tend to be competitive. Therefore, it is necessary to keep a compromise between these requirements when designing a motion controller for the joint of a humanoid robot.

Controller design is comprised of different stages: analysis, implementation, tuning, etc. As mentioned above, in the Rh-1 humanoid robot a distributed motion control philosophy will be implemented. In this philosophy, each joint is controlled independently by an intelligent motion controller with an implemented cascade control scheme. A PIV controller (explained in the chapter 3) basically combines a position loop with a velocity loop. The scheme implemented in order to control each joint is shown in figure 4.12. (This scheme doesn't contain a feed forward loop which will be added later by hardware and automatically tuned in order to achieve better error tracking).

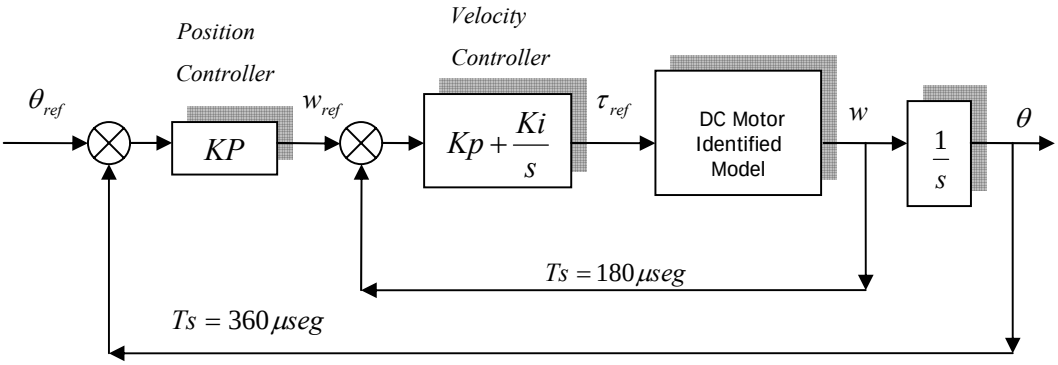


Fig. 4.12: Cascade control scheme implemented in Rh-1 humanoid robot

Sample time Ts was established by hardware limits and satisfies the Nyquist–Shannon sampling theorem:

$$Ts < \frac{1}{2 \cdot Fr} \tag{4.27}$$

where Fr is the highest movement frequency on the robot's links. Taking into account that the time of one step of the humanoid robot is not less than 0.5 seconds, the $Ts < 250\ ms$.

The fast internal loop of the presented cascade scheme contains a proportional and integral controller

$$\frac{Kp \cdot s + Ki}{s} \tag{4.28}$$

where Kp is a proportional gain, Ki is an integral gain.

The input for the PI block is a position error $e_{SPEED}(t)$ (internal velocity units of the implemented motion controller):

$$e_{SPEED}(t) = SpeedCommand(t) - SpeedFeedback(t) \tag{4.29}$$

The output of the speed controller for the next block, a current controller of the motor is:

$$I(t) = K_{Speed} \cdot \left(\int_0^t Ki \cdot e_{SPEED}(\tau) d\tau + Kp \cdot e_{SPEED}(t) \cdot p(t) \right) \tag{4.30}$$

where K_{SPEED} is a conversion factor from the D/A scale to the current.

Slowly position loop contains only a proportional gain and operates over the position error counted in counts of the encoder:

$$e_{POS}(t) = PosCommand(t) - PosFeedback(t) \tag{4.31}$$

and provides a command output for the speed controller:

$$CommandPI(t) = Kp^{out} \cdot e_{POS}(t) \tag{4.32}$$

This type of a control scheme has a reasonable advantage over a simple loop scheme. The internal, faster loop (velocity) absorbs the majority of disturbances until these affect the more important external loop (position).

For each joint a main variable to control is the angular position. Although it is logical to obtain a good functioning of the position loop, we need to achieve a good response in the velocity loop as well. To design a control, standard reference inputs are usually used. For the velocity loop design, a step input was used, while a ramp input was used for the position loop.

There are several parameters to take into account when designing the control: Rise Time, Peak Time, Overshoot and Setting Time (Figure 4.13).

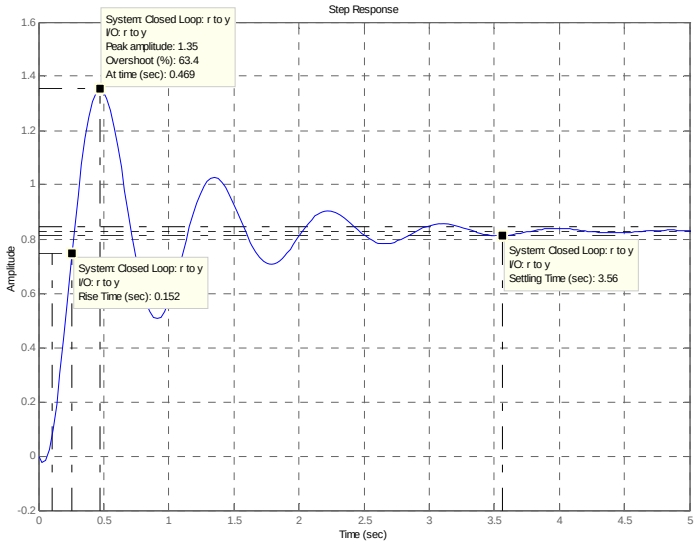


Fig. 4.13: Temporal output characteristics for step input

The basic characteristics of the transient response of closed-loop systems are related to its pole locations. The design consists of tuning the gain displacing closed-loop to the desired position. This way, a design turns into an adequate gain value selection. A root-locus method is a well known basic tool allowing graphical roots representation of a closed-loop transfer function.

The complex mechanical structure of a humanoid robot and the absence of damper mechanisms lead us to presume that disturbances will play an important role in joint control. As the disturbances we understand, every influences modifying the system's output and not included into the model. Turning back to the system identification, we should note that the worst model to identify the system was the ARMAX, which usually gave a good result in the case of disturbances acting on the system's input. Therefore we can infer that these disturbances affect the system's output (figure 4.14).

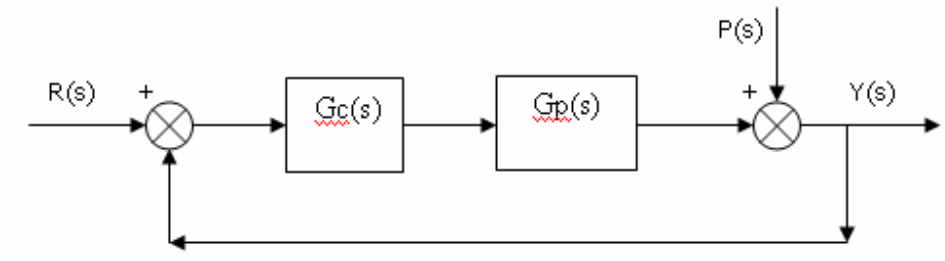


Fig. 4.14: Closed-loop system model

In figure 4.14, $Gc(s)$ is the transfer function of the controller (velocity or position) and $Gp(s)$ is the transfer function of the identified model for velocity controller design and the closed velocity loop for the position controller design. This system is linear, therefore, the superposition principle is permitted. The output is compounded by two components, $Y_r(t)$ - reaction to the reference input and $Y_p(t)$ - reaction to the disturbance:

$$Y(t) = Y_r(t) + Y_p(t) \tag{4.33}$$

Therefore, relation between the input and output is denominated by a transfer function:

$$\frac{Y_R(s)}{R(s)} = \frac{Gc(s) \cdot Gp(s)}{1 + Gc(s) \cdot Gp(s)} \tag{4.34}$$

While the relation between the output and the disturbance is:

$$\frac{Y_P(s)}{P(s)} = \frac{1}{1 + Gc(s) \cdot Gp(s)} \tag{4.35}$$

It is easy to observe that if $Gc(s)$ is sufficiently large, it reduces the influence of disturbances. On the other hand, the transient response for the disturbance should be the same as for the model input (current) because they have the same poles. Increasing the controller gain, we introduce

positive effects like the removal of disturbances and system exactness. But the gain can't be infinitely large because of an instability problem. Moreover, the control signal cannot be very big because it can saturate power converters, thus deteriorating motor behaviour.

4.4.2. Velocity Control

The velocity PI controller is characterized by a transfer function:

$$Gc(s) = \frac{Kp \cdot s + Ki}{s} = \frac{Kp \cdot (s + z2)}{s} \quad , \quad z2 = \frac{Ki}{Kp} \tag{4.36}$$

The main characteristic of a PI controller is an infinite gain at zero frequency that improves the exactness of the system. However, it provokes the instability of the system, so Kp and Ki values should be estimated accurately in order to provide a good transient response to a step input (without a big overshoot).

Figure 4.15 shows one of the root-locus (closed-loop poles – rose points, velocity controller pole and zero – red points) and temporal responses (reference – blue line and disturbances – red line) obtained during the velocity controller design. The model of the system is the ARX220 described in the previous section.

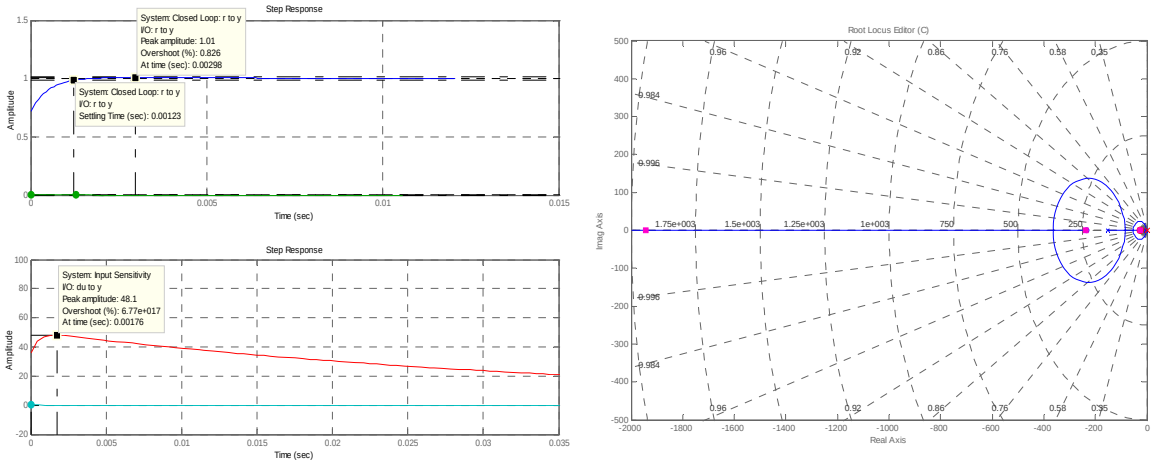


Fig. 4.15: Velocity controller design (Z2=-25, K=0.02)

Approximating the zero to the origin allows increasing controller's gain, and the response improves. However, it makes the system's response to disturbances become slowly. As it is an important condition for good motor behaviour, the error should be eliminated as quickly as possible.

Figure 4.16 shows experimental results applying a developed controller to joint 10 of the humanoid robot Rh-1. The blue line is command velocity, and the red line is real velocity. Sample time is 360 μs and velocity command frequency is 10Hz.

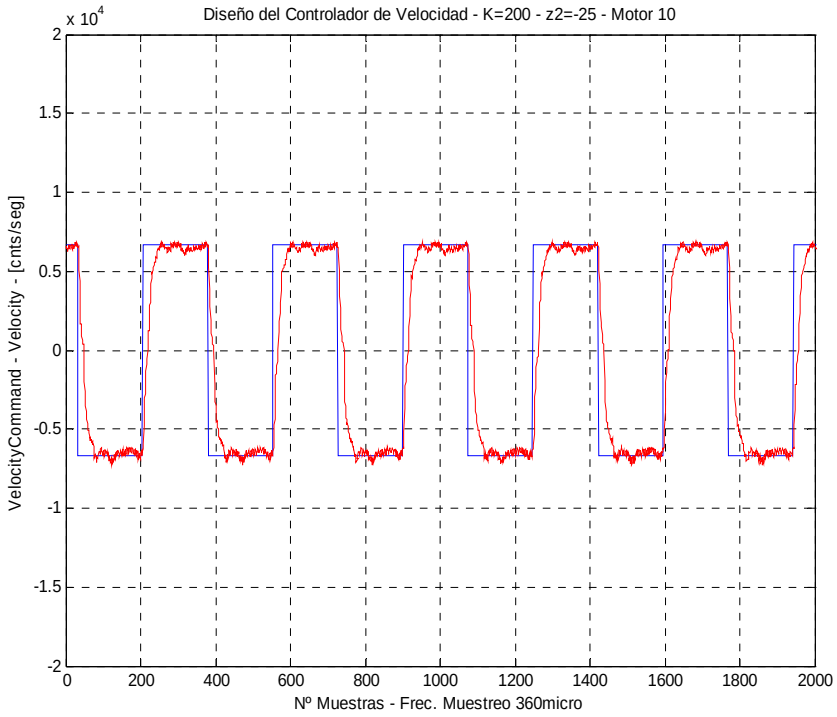


Fig. 4.16: Experimental results for the motor 10

The results obtained are good, but it is necessary to take into account the final design objective – obtain a good system response for the position loop.

4.4.3. Position Control

The position loop is external and slow and includes only a proportional gain. The proportional gain increases a total system gain, while the phase diagram remains the same. Position control was introduced in order to correct a transient system response. Generally, it is tuned in the same way as a proportional gain of velocity controller. By increasing the gain, we decrease the influence of disturbances (also decreasing the sensibility of the system), and the transient response becomes faster and without notable oscillations. Evidently, there is a limit because it is not possible to increase the gain infinitely, as it affects the system stability.

The position controller slightly modifies reference velocity in order to compensate for its regulation error, but the position error is mainly determined by the velocity controller.

The experiments conducted show rather good results. These were realized using a ramp reference position input. The trajectory point-to-point, between -1000 and 1000 encoder counts, was executed periodically with a 0.3 s period. Figure 4.17(a) shows command (blue line) and real (red line) velocities. Figure 4.17(b) shows a position error for the same experiment. The position error is maintained at nearly 0 during the trajectory and does not exceed 35 counts when the trajectory is changed that can be considered as a good result. Undoubtedly, if the velocity response is bad, it is impossible to obtain a good position control and vice versa.

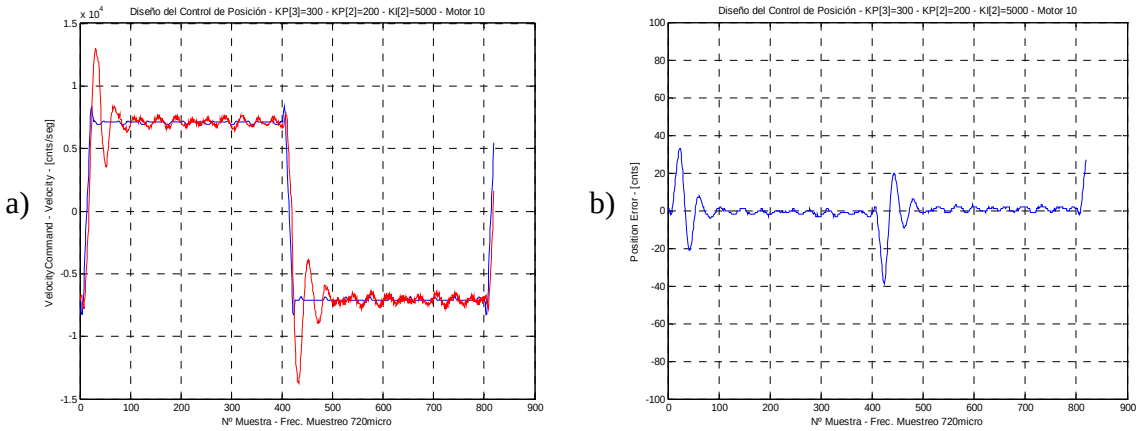


Fig. 4.17: Experimental results for the motor 10. a) Real velocity/command velocity b) position error

As mentioned above, velocity response mainly determines a position response. When a velocity controller is slow, position control is non effective. When the velocity response is faster, the error always stays inside acceptable margins always when the velocity peaks are not too high.

In order to study the reliability of the designed control, different experiments were carried out. Figure 4.18 shows different postures (positions of the joint 10) of the humanoid robot Rh-1 and figure 4.19 shows position errors in these postures.

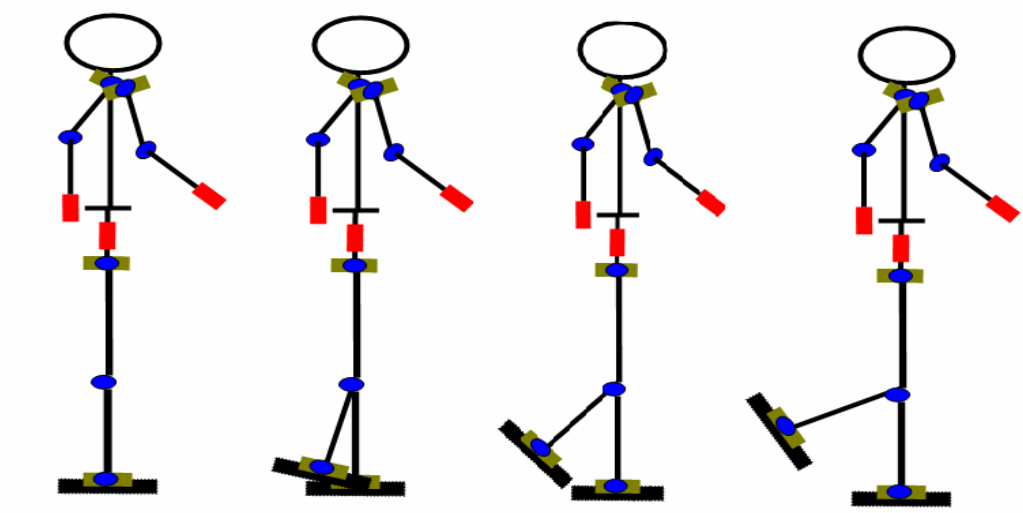


Fig. 4.18: Different positions of the joint 10 in counts of the encoder a) 0 counts b) -50000 counts c) -100000 counts d) -200000 counts

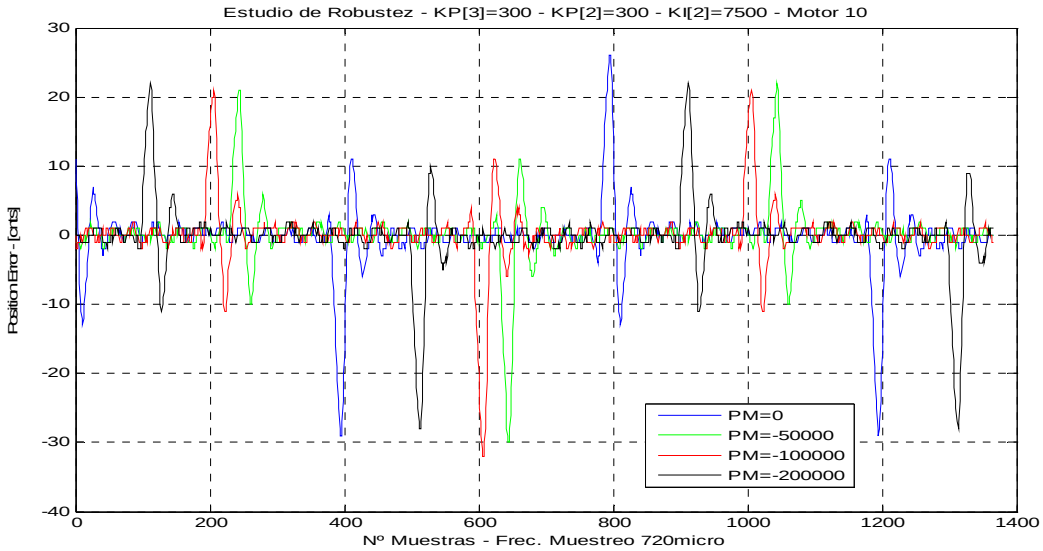


Fig. 4.19: Joint 10 position error

It is observed that when the designed control system is applied, the position error is almost the same for different walking postures of the humanoid robot. This led us to undertake a more complex study of the necessity of an adaptive control for humanoid robot joints involved in bipedal locomotion.

4.5 Adaptive control

This section provides a study of the adaptive control for humanoid robot joints. Firstly, we need to answer the question: Is it really necessary to implement adaptive joint control for humanoid robot walking?

4.5.1. Load Torque

As mentioned above, a motor's load torque T_L appears due to the mass of a joined link, and its value depends on the space position of the link. As the position of the link changes while the humanoid robot is walking, it can be considered a time variable parameter. In order to calculate joint load torque and the moment of inertia, it is possible to represent the link by static and dynamic equivalent mass.

$$M = \sum_{i=1}^{n} m_i \tag{4.37}$$

$$M \cdot R_{cdg} = \sum_{i=1}^{n} m_i \cdot r_i \tag{4.38}$$

$$M \cdot R_{cdg}^2 = \sum_{i=1}^{n} m_i \cdot r_i^2 \tag{4.39}$$

where m_i is mass and r_i is a vector of positions i the link segment.

Equations (4.37) and (4.38) show that the equivalent mass M has the same Centre of Gravity (COG) as the link, therefore, they are statically equivalent. The dynamics (motion around the COG) will be the equivalent adding the equation (4.39), there $M \cdot R_{cdg}^2$ is the moment of inertia of the link with respect to its COG.

The following equation shows how the equivalent load torque depends on the position of the equivalent centre of mass:

$$T_L = M \cdot g \cdot l \cdot \sin(\theta) \tag{4.40}$$

where M is the total mass of the link (constant), g - gravitational acceleration, l - distance between the link's COG and motor axis in the motion plane, θ - angle between the line connecting the COG with the motor axis and the weight vector of the link. As the relative position of the COG with respect to the motor axis is the function of the link space position, l and θ will be variables.

Equation (4.40) describes the load torque of the link, but it differs from the torque affecting the motor. Every link is joined with the motor by a reduction gear (Harmonic Drive + belt), therefore, the gear ratio should be considered. In an ideal case, considering an ideal reduction gear (without friction), the energy should be conserved:

$$\Omega_{motor} \cdot \tau_{motor} = \Omega_{link} \cdot \tau_{link} \tag{4.41}$$

where Ω_{motor} and Ω_{link} are positions, τ_{motor} and τ_{link} are load torques affecting the motor and the link respectively.

$$\frac{\Omega_{motor}}{\Omega_{link}} = r \tag{4.42}$$

where r is gear ratio. For the Rh-1 humanoid robot $r \approx 160 \cdot 4 = 640$, therefore,

$$\tau_{motor} = \frac{\tau_{link}}{r} \tag{4.43}$$

That is, the load torque in the motor side is $r = 640$ times smaller then the load torque of the whole link. Thus, every load torque change in the link will be attenuated on the motor's side and, in an ideal case, could be considered as a constant. In reality, every experiment demonstrates that the load torque on the motor side shows a small variation and slightly modifies the dynamics of the system. This effect is more pronounced in positions with higher load torque (higher gradient of the torque).

Some experiments were carried out with motor 9 of the humanoid robot Rh-1 in order to observe the motor's behavior upon application of different load torques. Motor 9 moves the left hip, therefore, the associated link is characterized by the most significant mass and relative movement. In every experiment, the moment of inertia of the link was maintained constant. Thus, every change we observed in the system response is only because of the change in the load torque. The posture of the link in every experiment was the same (extended), with the foot almost perpendicular to the leg (Figure 4.20). The gradient of the load torque for these experiments was probably even bigger than in the real humanoid robot walking movement.

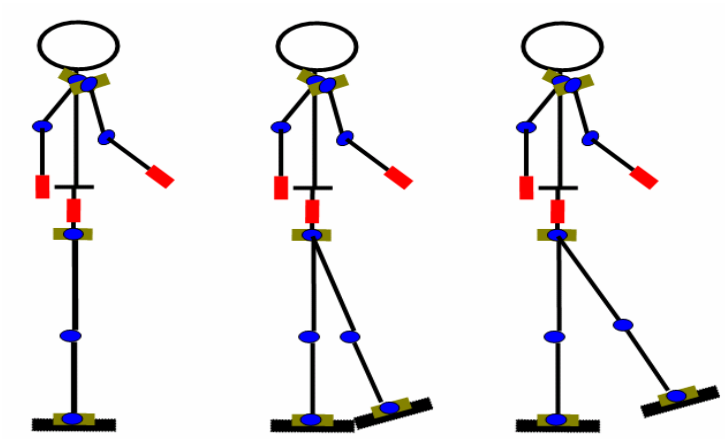


Fig. 4.20: Different positions of the link. a) PM=0º b) PM=15º c) PM=30º

From simple observation of the results (Figure 4.21(a), it is clear that the motor's response to the step signal is affected by the load torque. Output gain in particular decreases when the load torque increases. Logically, when for the same level of torque (current) the resistance rises, acceleration diminishes as well as the maximum velocity reached.

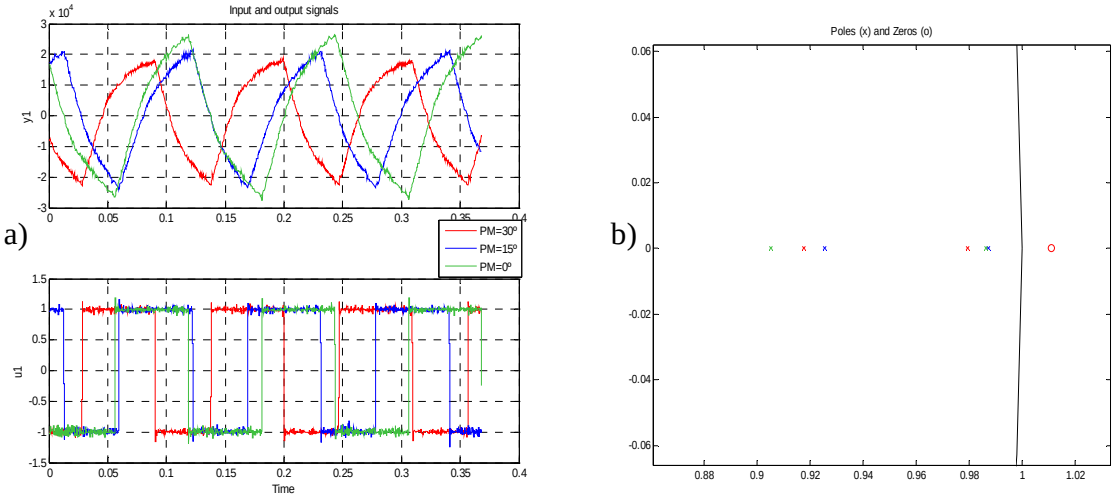


Fig. 4.21: Load torque influence experiments a) Velocity output and current input b) Root-locus of the system

The root-locus diagram (Figure 21(b)) shows that the dynamic is also slightly different, because of the slight variation of the torque in each experiment.

4.5.2. Moment of Inertia

The moment of inertia J is other parameter of the model. It also depends on the posture of the link joined to the motor, therefore, its value changes during humanoid robot walking. It is necessary to consider this parameter and study its influence on the control.

In general, the moment of inertia of a body with mass with respect to any axis AA' is defined as the integral:

$$I = \int r^2 \cdot dm \tag{4.44}$$

where r is the distance between the element of the mass dm and the AA' axis.

If Z is an arbitrary axis and Zc is a parallel axis that passes through the C.O.G. of the body, the moment of inertia with respect to the parallel axis is (by the Steiner theorem):

$$I = Ic + M \cdot a^2 \tag{4.45}$$

where I and Ic are moments of a body's inertia with respect to the Z and Zc axis respectively and M - body mass.

The effect that produces the Moment of Inertia on the dynamics of the system is clearly seen if we consider a manipulator moving on the plane parallel to the ground. In this hypothetical case, the counteraction to velocity changes depends only on the mass distribution of the manipulator with respect to the axis of rotation. That is, when the manipulator is more extended, it is more difficult to accelerate and the motor's response is slower. In this case, the moment of inertia is larger.

When the link is composed of at least two parts and relative movements (or variation of distribution of masses) is possible between them, we can say that the link can change its moment of inertia. Of the humanoid robot Rh-1's 21 DOF, only 12 can modify its moment of inertia when it moves (walks). The moment of inertia total is the sum of the moments of inertia for each part of the link with respect to the axis of the considered motor. In the previous section J represented the moment of inertia affecting the motor axis (rotor+link). The inertia of the rotor is very small with respect to the inertia of the link (because of the difference between masses and distances). Therefore, using J as the moment of inertia of the system, we are referring only to the moment of inertia of the link.

The moment of inertia affecting the motor axis decreases with respect to the moment of inertia of the entire link by the following equation:

$$J_{MOTOR} = \frac{J_{LINK}}{r^2} \tag{4.46}$$

where J_{MOTOR} - the moment of inertia affecting the motor, J_{LINK} - the moment of inertia of the link and r - gear ratio (for the humanoid robot Rh-1 $r = 640$). As a consequence of the square of reduction, attenuation is very high, and the moment of inertia is strongly reduced on the motor side. Theoretically, the moment of inertia should directly affect the dynamics of the system, i.e. the rapidity of the system response to the changes in its input. Thus, determining the position of the poles of the transfer function $\Omega(s)/I_a(s)$ and $\Omega(s)/T_L(s)$, we are able to determine its transient response.

In order to ascertain the influence of the moment of inertia on the dynamical model (Equation 4.20) some simulations were performed. The values we used for these simulations are not real, but were selected in order to show how system response differs when the moment of inertia of the link changes.

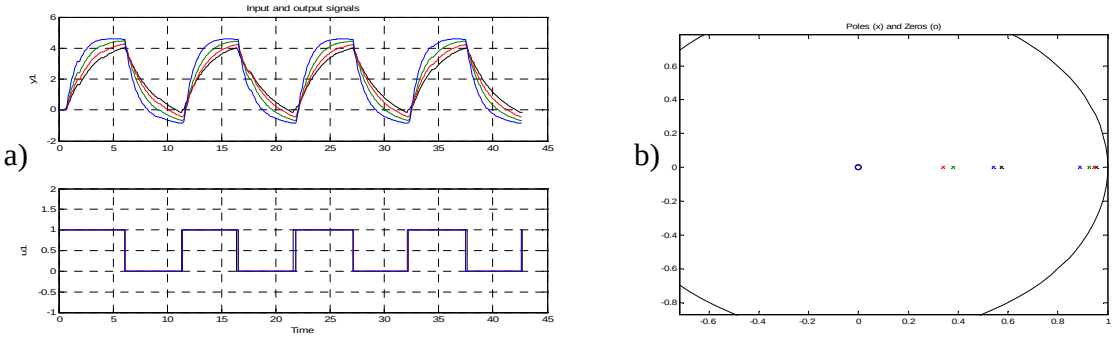
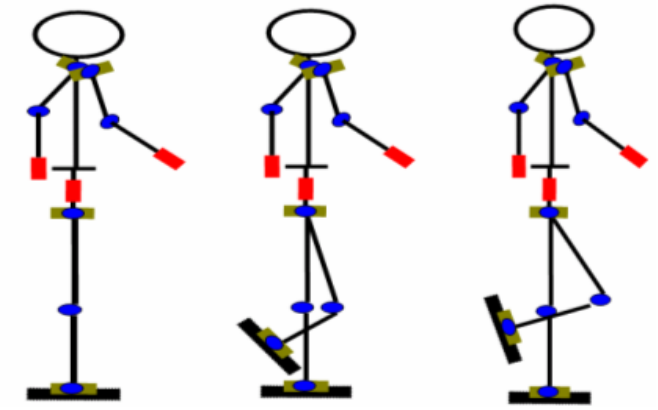


Fig. 4.22: Simulation of influence of the moment of inertia a) Velocity output and current input b) Root-locus of the system

It is observed that when the moment of inertia is greater, the system response to a step current input or any disturbance is slowly (figure 4.22(a)), therefore, the dominant pole is located near the unitary circle. Changes in the moment of inertia do not affect the gain of the system.

Experiments carried out in this study consisted of observing system responses for different postures (moments of inertia) of the link. The same as with that the load torque in the previous study, the moment of inertia should be the only variable of the system. This is not a simple task with a humanoid robot. When the motion is produced in a plane different from the horizontal one, each change in the link's posture modifies the load torque, hiding the real cause of changes in motor behaviour. But by using equations (4.37), (4.38) and (4.39), different postures with constant equivalent load torque can be obtained, thus, always maintaining the link's COG in the same position during the experiment. The magnitude $l \cdot sen(\theta)$ should be maintained as a constant.

For the same reason as with the load torque, changes in the moment of inertia mostly affect the humanoid robot's hip motors motions. Changes to the moment of inertia in the experimental postures shown in the figure 4.23 are more extreme than those of a real humanoid robot walking.



a) b) c)

Fig. 4.23: Different positions of the link a) J max b) J med c) J min

Observing the experimental results (Figure 4.24(a)), we can conclude that changes in the dynamics of the system due to change in the moment of inertia in different postures of the link are very small.

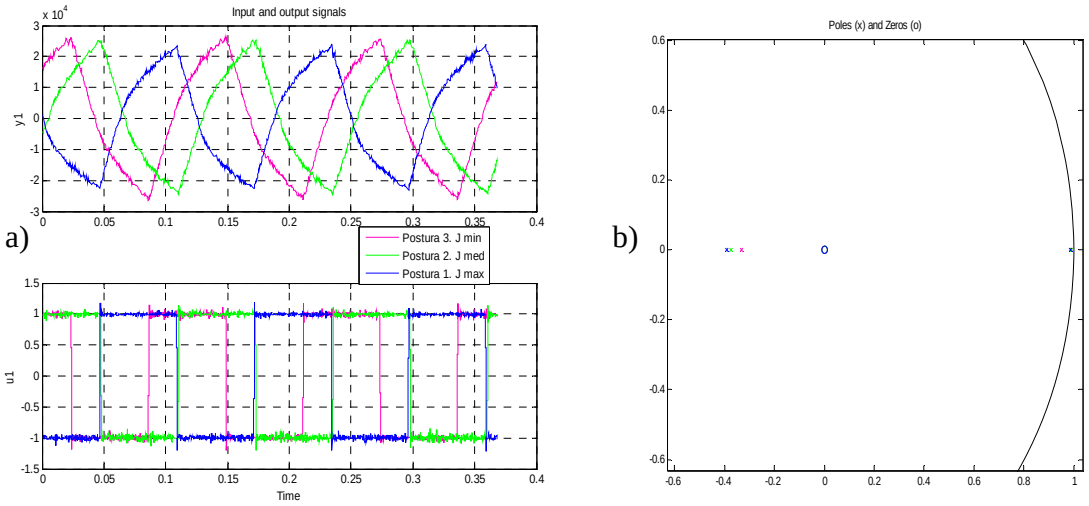


Fig. 4.24: Moment of inertia influence experiments a) Velocity output and current input b) Root-locus of the system

In the root-locus diagram (Figure 4.24(b)) it is evident that the position of the dominant pole is the same in every case. Whereas the second pole, controlling the fast dynamics of the system approaches the origin when the value of the moment of inertia is smaller. In practice, these variations are insignificant.

4.5.3. Implementation

The basic idea of joint adaptive control for a humanoid robot is to modify in real time the parameters of the control in accordance with the instant system behaviour. After obtaining experimental results, we can make some conclusions about the control system to be implemented

for joint control of a humanoid robot. It was shown that basically the load torque affects the control gain of the system the same way the moment of inertia affects the dynamics of the system. Therefore, we should provide a study of how the adaptive control mechanism could improve the behaviour of the joint control system for humanoid robot walking.

Basically, there are two ways to implement an adaptive control: direct and indirect control schemes. The indirect control algorithm requires a dynamical model of the system that involves its online identification. There are three main problems:

1) In the event of strong disturbances, the identification works poorly. System's velocity response is impure and estimation uncertainty increases.

2) Online control requires closed loop identification. The problem is that closed loop data has less information than the open loop system. Conventional methods can't be used, and estimation becomes extremely complicated.

3) The identification has very high computational and communications costs. Taking into account a high number of controlled DOFs participating in humanoid robot walking, it is extremely complicated to realize online motion control of all joints.

On the other hand, the direct adaptive control seems to be the best choice for the online solution. The main difficulty is in selecting an adaptation algorithm able to provide the stability and the correct system functioning, as these schemes are normally characterize by unreliability.

Therefore, the simplest adaptive control that can be implemented in the humanoid robot is the gain scheduling adaptive control algorithm (Figure 4.25). It includes an offline direct controller design with online gain scheduling. It doesn't need online change of system identification and controller design processes described in previous sections.

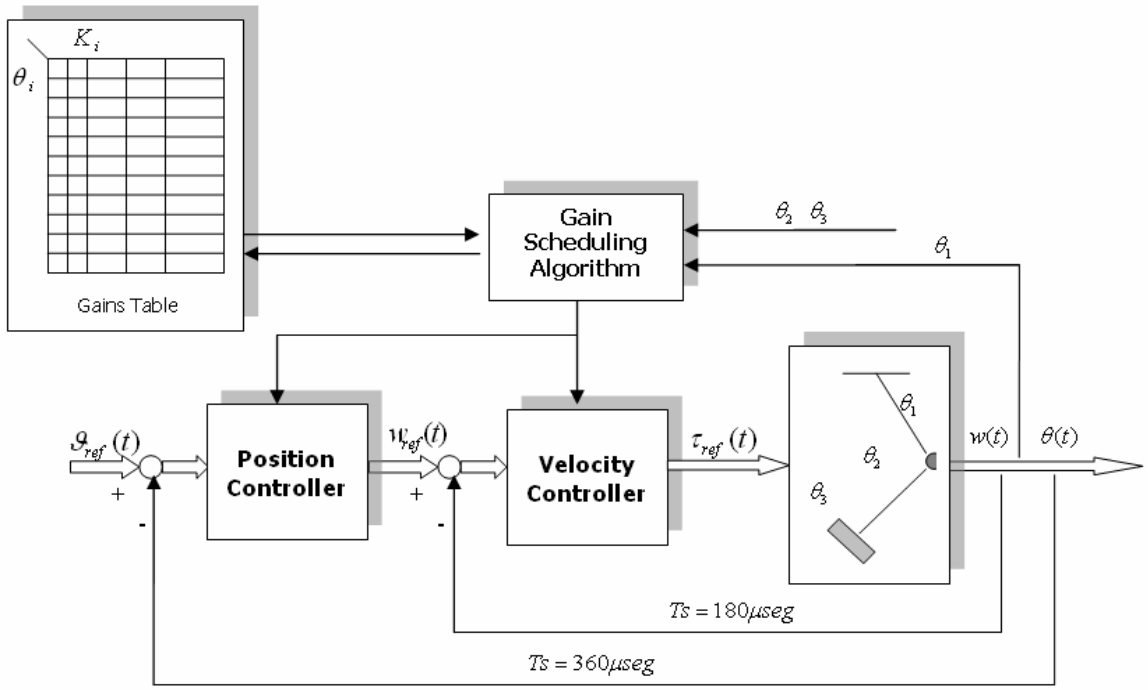


Fig. 4.25: Gain scheduling control algorithm

In this control scheme, a number of controller's gains are previously calculated for different functioning conditions (postures). The gain scheduling algorithm enables online the gain corresponding to the actual humanoid robot posture. The implementation of the adaptive control is not a trivial task and involves some risk and limitations. First of all, a link is composed of several parts, for example for motors 4 and 9 corresponding to hips of the Rh-1 humanoid robot, there are 3 parts composing the link (hip, ankle and foot). This suggests that, in this case, a 3 DOF should be considered to establish different working conditions (postures). This produces a huge number of functioning points. For example, if the range of movement for a link is 90º sampled into 15º segments, it is necessary to consider $\left(\frac{90}{15}\right)^3 = 216$ points (number of controllers to design).

After the controller was designed, some experiments comparing adaptive and conventional joint control were carried out. In these experiments, the joint 9 of the Rh-1 robot was moved (step signal) to different positions with different load torques and moments of inertia. Figures 4.26 and 4.27 show one of the typical comparison results between a conventional cascade controller and the implementation of the adaptive gain scheduling controller.

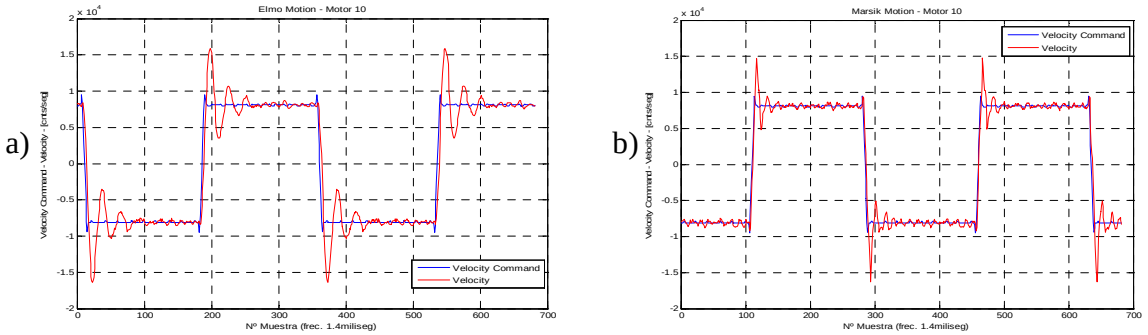


Fig. 4.26: Velocity Command/Real a) Cascade Controller with constant parameters b) Gain scheduling controller

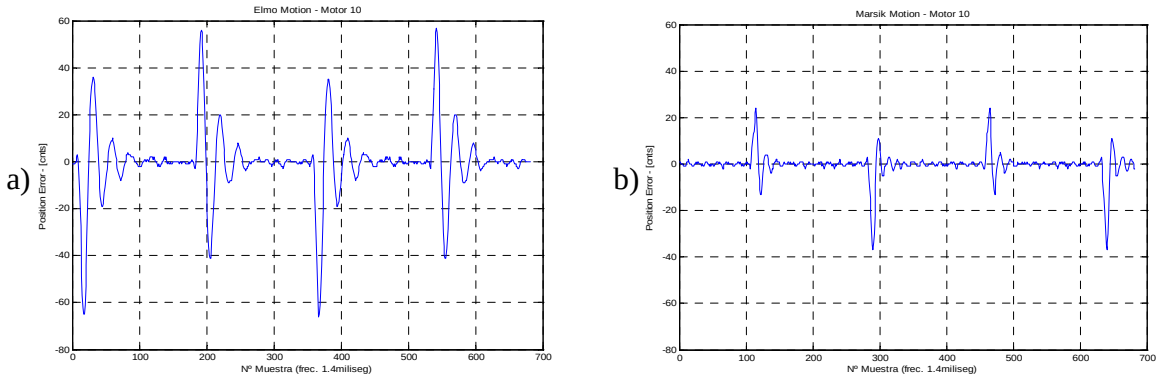


Fig. 4.27: Position Error a) Cascade Controller with constant parameters b) Gain scheduling controller

In figures 4.26 and 4.27 it is observed that the gain scheduling controller has better performance. It has less overshoot and smaller setting time in the velocity control loop and smaller position error in the position control loop. Evidently, this occurs because the adaptive controller has optimal gains for each possible joint's position (posture), taking into account the possible changes in the moment of inertia and load torque, while the conventional controller has the same control gains for every posture.

On the other hand, these experiments also show that for the real humanoid robot in walking mode, the influence of disturbing parameters (load torque and the moment of inertia) is not so large as to adopt the adaptive control in function of these variables. For example, the position error decreases about 40 motor counts that in terms of joint angles is 0.012°. This error in joint positioning can be negligible.

At the same time, one of the adaptive control's (Gain scheduling or other algorithm) main problems is the need for its online implementation. When the reference gain table is created, the implementation of the algorithm is simple. However, this algorithm will run in parallel with other control procedures, such as body attitude or foot control, gait generation, etc., thus capturing computational and communicational resources from the essential humanoid robot walking control.

In conclusion, in order to decide the need of the adaptive control for every specific case, it is necessary to gauge the benefits and inconveniences of such a control system in different working conditions. If the benefit is small, it is better to reject the adaptive control algorithm and rely on the conventional controller. Moreover, when a humanoid robot walks, angular position of articulations implicated in walking changes slightly. For example, the hip (motor 9) angular position of the humanoid robot Rh-1 varies only 90º in both directions. In this trajectory, the influence of disturbance parameters, such as load torque or the moment of inertia, doesn't change radically (as was shown in presented experiments) and can be regulated by a conventional controller. By this reasoning, and relying on the developed in previous section PI controller (experimental results show good error compensation for every walking posture) we rejected the implementation of adaptive control algorithm for joint control of the humanoid robot Rh-1.

4.6 Conclusions

As shown, the joint control problem can be solved via different modes. In this research, a dynamical model of the system and a classical control theory were used. The identification of the dynamical model is not a simple task. The basic problem is that there are more signals (load torque, vibration, etc.) affecting the system's output than we can measure. Thus, these signals were considered as disturbances, and in the experiments carried out, we tried to minimize the influence of these terms using short trajectories and strong input signals.

Identification was made in the open loop. Open loop identification has the inconvenience that the system is identified in unusual working conditions (usual working condition is a closed loop control), but with careful design of the experiment, the obtained results are useful. Another issue is the validity of the system dynamical model obtained, but knowledge all of its limitations made it possible to design a good controller.

Another aspect considered was the influence of the reduction ratio to the dynamical model. The redaction mechanism plays an important role in the dynamics of the system. On the one

hand, it decreases the interdependency between articulations that was used to select a stand alone decoupled joint model. It also attenuates the dynamics of the link, simplifying the control. On the other hand, it introduces non-modeled friction and hysteresis. These non-lineal effects should affect the system's behavior and are not considered by the model. However, the estimated model response fits very well the real motor response, confirming the insignificance of these effects. In order to execute identification online, a LS (Least Squares) algorithm for the ARX structure model was implemented.

After estimating the dynamical model, the design of the control system was carried out. The main objective was to obtain a fast and precise position response. The implemented cascade control scheme requires tuning at the first velocity loop and then tuning the position control loop. Good tuning of the velocity control loop is a fundamental condition for proper functioning of the entire control system because the internal, faster loop (velocity) absorbs the majority of disturbances until these affect the more important external loop (position). Finally, the design of the velocity loop consisted in the control design for a second order system there conventional PI controller shows perfect results.

In the humanoid robot, the overshoot in the motor's response is not permitted because it can provoke a collision of the robot with the environment. However, motor velocity profiles for bipedal walking are very abrupt with maximal velocities close to the motor's physical limits (current or velocity). Fast control demonstrating some overshoots can generate unacceptable peaks of velocity and change the desired joint trajectory. Thus, it is necessary to obtain a dampened response for the velocity and position loop.

It is also necessary to consider practical aspects of the controller design. For example, amplifiers or electronic devices providing power to the motor have their physical limitations. When the output signal of the amplifier is proportional to the reference signal, this proportionality is valid only between minimal and maximal values. Exceeding the maximal value, the output signal remains constant and does not depend on the input reference. The presence of error and the use of the integral part of the PI controller can provoke the signal to exceed the limits of the lineal behaviour of the amplifier and remains constant independent of the system error at the moment. Therefore, control gains can't take high values. This saturation effect was taken into account in the model used for the controller design.

The velocity and position controller also have some elements such as high pass filters and feed forward factors. The speed feed forward based on knowledge of the model of the joint can't improve system behaviour in the case of disturbances and is implemented in order to eliminate tracking errors at a steady speed.

Velocity and position control design was developed in two stages. At first, optimal gains were estimated using the identified model and the root-locus designing method. Then, as the mathematical model in practice always slightly differs from the real system, a fine tuning was performed experimentally.

Also, a profound study of the influence of the load torque and the moment of inertia of the link on the system's behavior was carried out. Experiments, performed with the motors of hips (probably mostly affected by the posture changes motors), show that the system response doesn't vary appreciably. There are two possible reasons. First of all, the attenuation of the system's dynamics (moment of inertia) produced by the reduction gear is very high. And secondly, the designed controller's cascade configuration reduces the disturbance (load torque) effect in the

internal faster current and velocity loops and doesn't allow it to affect the principal control variable (position).

In conclusion, the adaptive gain-scheduling controller was designed. It shows better control performance than the standard static parameters controller. Nevertheless, taking into account its high implementation and computational costs, it was confirmed that there is no necessity for joint adaptive control for humanoid robot biped walking. The use of the adaptive control can be justified in the event of strong disturbances (a possible situation when the mechanical structure of the humanoid robot is not carefully designed and assembled).

The joint control system obtained in this research is the core of the global motion control system of the humanoid robot. Walking experiments, presented in the chapter 7, will prove that the Rh-1 robot provided with a designed joint motion control system can walk smoothly.

Chapter 5

Stabilization control of a humanoid robot

CHAPTER 5

Stabilization control of a humanoid robot

5.1 Introduction

Humans usually walk using a specific walking pattern in usual environments. In cases of unexpected disturbances this pattern can be changed immediately to another in order to adapt to the requirements of the terrain. The humanoid robot operates in the same space as the human and needs the same mechanism to adapt its walking to the changing conditions. Moreover, the walking pattern alone cannot guarantee the stability of the walking humanoid robot even on a flat and ideal terrain because of the dynamical constraints of the artificial body. Therefore the controller, which is able to stabilize the robot during bipedal walking, is an essential part of the motion control architecture of any humanoid. The following sections set out the theoretical issues and a detailed design of the new stabilization system for humanoid robots.

Firstly, it is necessary to consider theoretical aspects of humanoid locomotion. It includes a study of a humanoid robot walking (ZMP concept, Inverted pendulum mode, trajectory generation and motion planning). After the safe motion pattern is created, the reference motion is transmitted to the mechanical system of a humanoid robot. A humanoid robot walking with two legs is not a dynamically stable system. During the trajectory execution, lateral accelerations and disturbances can cause the robot to fall over. The next part of the work will consist of the theoretical proving of the need for stabilization control and choosing the easiest way to implement it. Basic questions in this context are: How can we control stability? What is necessary to change in the humanoid robot motion to achieve stability? On which part of the robot should we act in order to stabilize it? Is it possible to realize a decoupled control of the complex dynamics of the robot in order to achieve walking stability?

After these questions have been solved, the control structure (architecture) will be developed. It consists of two principal parts: Body attitude control and ZMP control. The research work in this chapter answers the question of how the dynamics of the humanoid may be decoupled into two independent controls. The next step consisted of the modeling of the dynamical system and the development of the system identification for both of the proposed control algorithms. After that, a detailed discussion of the complimentary systems (sensorial data acquisition and processing) needed for the practical realization of the Stabilizer is proposed. After the Stabilizer was designed, the control algorithms were implemented as real-time plugins for an OpenHRP simulator (which later can easily be used for the real HRP-2 robot). The experimental part of this research work will be presented in details in the chapter 7. It consisted in experiments with an OpenHRP simulator executing characteristic joints motions and registering the required system's

response. Finally, it will be demonstrated that the model of the HRP-2 humanoid robot provided with the designed stabilization system can walk stably.

5.2 Description of the humanoid robot walking

5.2.1 Bipedal locomotion characteristics

The synthesis of the artificial bipedal locomotion is a very complex problem. Human beings possess the ideal bipedal locomotion combining active and energy-efficient passive phases in which the kinetic energy of the previous step is used to generate the next one. Therefore the best way to produce such a type of motion for a walking machine is to copy human motion.

Most of human walking is automated motion (passive) and does not use the central nervous system during the continuous walking mode. By this, the locomotion process is similar to the repetitive algorithm executing permanently until such time as some perturbations are detected. In human locomotion, spontaneous motion due to the redistribution of efforts throughout the muscular system modifies relationships between forces bringing these relationships into the equilibrium or otherwise, bringing these out of it. The study of these systems and its motions requires some simplifications because legged locomotion systems and particularly an anthropomorphic legged mechanism is an extremely complex system from the mechanical and control points of view.

There are about 350 pairs of muscles in the human body, and they are used to carry out a complete walking activity. This system represents a very high dynamical complexity. Mathematical analyses of this motion indicate that this interaction does not have a sole dependence. It occurs because the relationship between the force and the motion is generated in the biomechanical sense based on the second order differential equations whose solution requires two initial conditions. These integration constants (initial position, initial velocity, etc.) can lead to absolutely diverse effects during the same initial condition. This complexity and restrictions posed by the mechanical system, actual actuators and motion control systems state that the exact copying of human motion seems to be unrealizable.

However, current research on passive (biomechanics approach) bipedal locomotion gives some results. Sustained efforts have been made in the making of controlled machines that mimic human gait and some forms of animal locomotion. However, reports of much lower levels of neuro-muscular controls in simple human gait have appeared in the recent past, which suggest that a large part of the walking motion could be simply passive. Analysis of passive walking is fairly recent. In fact, passive locomotion caught the imagination of analysts only after Ted McGeer [McGeer 90] built a passive walker around 1990 and showed that his two-legged walker could reproduce stable gait without any controls.

The most progress was revealed in the active bipedal locomotion [Hirai 98], [Kaneko 2002], [Park 2005]. This type of the locomotion is developed and implemented as artificial human – like bipedal motion based on the previous planning of each step and the real-time automatic control of its execution and is the topic of this research work.

The basic characteristic of the locomotion mechanism, especially the bipedal one, is the presence of the degree of freedom which is not controllable by any actuator at the contact point of a robot's leg and the ground. This DOF cannot be controlled directly. It is necessary to realize

the appropriate motion of the entire system (of all DOFs), otherwise, the system can tilt with respect to the edge of the foot and fall over. Thus, the influence of this non-controllable degree of freedom is crucial for the system stability.

Another fundamental characteristic of the legged locomotion is the repeatability of the motion. For bipedal locomotion the period of repetition of motion is one step (gait). It supposes that position and velocity at the beginning and the end of each step have to be the same. By this means, an additional constraint can be added in order to find the possible solutions of the motion synthesis.

The third characteristic of bipedal locomotion is the permanent change of the situation when the mechanism is supported by one foot (single support phase) and when both feet are in contact with the ground (the double support phase). The second situation is statically stable and there are no additional moments affecting the robot. The first situation is statically unstable because when one foot is on the ground, the other is transferred from the back to the front position producing lateral accelerations affecting the mechanism. Thus, the locomotion mechanism changes its structure during a single walking cycle from an open to a closed kinematics chain. Each of these two cases present different dynamical situations and should be taken into account in artificial gait synthesis and control.

The phenomenon of locomotion can be understood more easily if we study the motion of a human in three basic planes: sagittal, transversal and frontal (Fig. 5.1). It is important to mention that the most important motions occur in the sagittal plane because it coincides with the main walking direction. Motions in transversal and frontal planes are not so radical. The joints responsible for motion in these two plains (for the most part in the frontal plane) help significantly in the stabilization of the locomotion cycle.

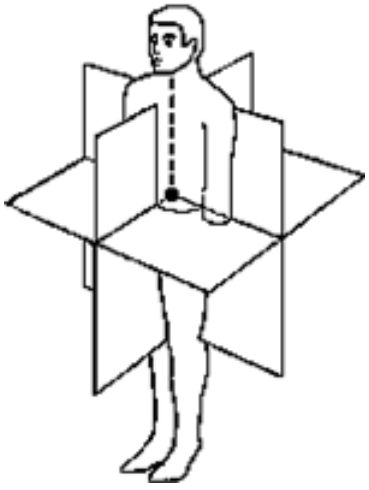


Fig. 5.1: Motion planes

Figure 5.2 shows a basic analysis of the walking cycle (forward walking gait) of bipedal locomotion. The fool cycle is divided into two, left and right steps. The left step starts when the right foot is touching the ground at the end of the previous step. At the beginning of the step, a mechanism should provide transfer of the weight from one foot to the other during the double support phase. After all the weight is concentrated on the right foot, the mechanism is able to

start lifting the free left foot. During a single support phase the left leg is transferred into a new position on the ground.

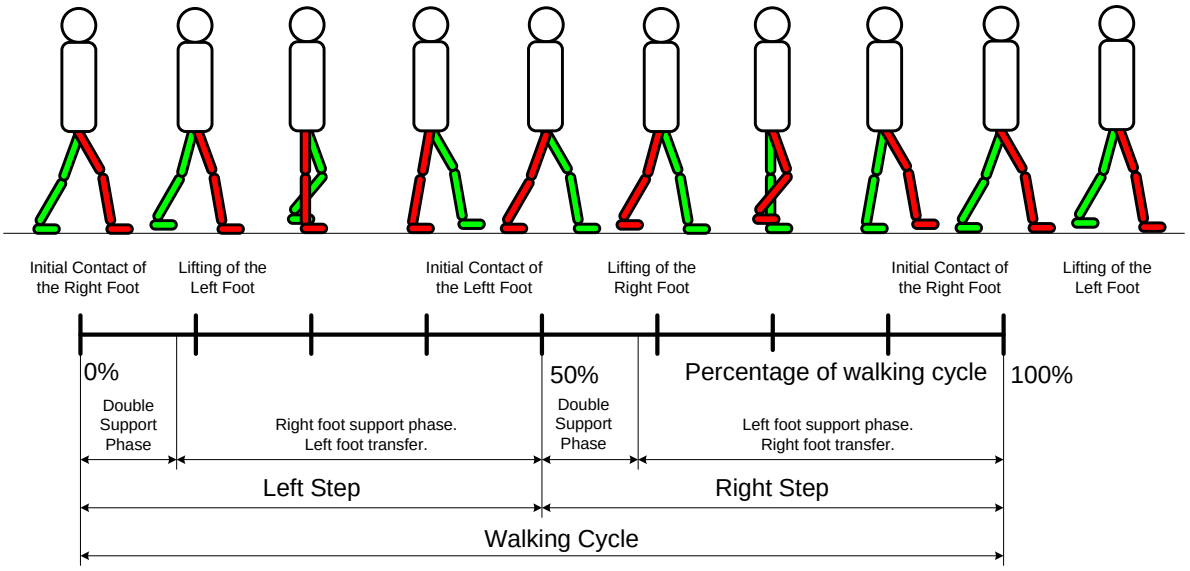


Fig.5.2: Walking Cycle

After the left leg touches the ground, the next (right) step with the same basic phases is started. After the right step, the whole cycle is repeated again and again until the motion has stopped or the gait has changed.

5.2.2 ZMP Concept

In an electro-mechanical system like that of a humanoid robot all motions are provided by electrical motors. To generate a motion of the robot means to provide trajectories for all joints implied in this motion. In this context the main question is what is the criterion which allows for safe and stable humanoid motion? During the motion generation of the artificial bipedal mechanism it is necessary to take into account many different characteristics. The most important of them is the Zero Moment Point (ZMP) concept introduced by Miomir Vukobratovic in 1969. It allows for a unique criterion to be applied for determining the stable bipedal locomotion of a humanoid robot. Because of its universality and simplicity, this concept is currently widely used as a basic method for generating bipedal locomotion. The ZMP is also indispensable for establishing unavoidable feedback with respect to the dynamics of the bipedal system and ground reaction forces.

The basic idea consists of the fact that if there is a non controllable directly DOF at the contact point between the foot and the ground, it should be controlled indirectly in order to ensure appropriate dynamics of the entire system. The overall indicator of the bipedal mechanism above the foot behavior is the point on the ground where the influence of all forces acting on the system can be replaced by one single force termed as Zero-Moment Point. To clarify this concept let us consider a supporting foot of a bipedal mechanism in the single support phase (fig. 5.3). So as not to consider the complex dynamics of the entire upper body of the mechanism we can replace

its influence by force F_A and moment M_A applied to the point A where the foot is connected with the leg. Also, the weight of the foot acts at its gravity center (point G). Finally, the contact of the foot with the ground produces its reaction force R at point P.

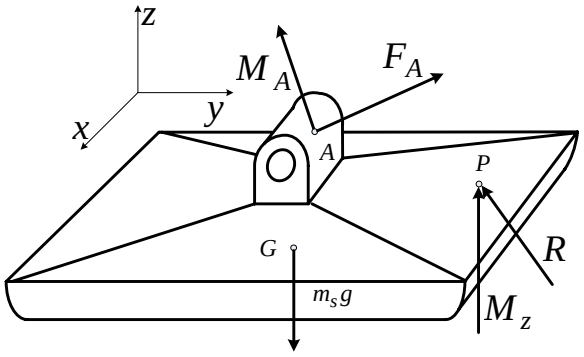


Fig. 5.3: Forces acting on the foot of the bipedal mechanism [Vukobratovic 2004]

In general, the total ground reaction consists of three components of the force R (R_x, R_y, R_z) and moment M (M_x, M_y, M_z). The friction force is acting at the point of contact of the foot with the ground. As the supporting foot is supposed to be immovable throughout the whole supporting phase (we assume that there is no sliding at the contact point) that means that the horizontal components of the reaction force R_x and R_y represent a friction and should balance the horizontal components of the body action force F_A. By the same reasoning, the vertical reaction moment M_z represents the moment of friction forces and should balance the vertical component of the body action moment M_A and the moment caused by horizontal components of the force F_A. The vertical reaction force R_z represents the ground reaction that balances the vertical forces.

But when we consider the horizontal components of the moment M_A, it can be compensated for only (due to the unidirectional nature of the connection between the foot and the ground) by changing the position of the reaction force R within the support polygon in order to balance an additional load. Figure 5.4 illustrates a planar case in the *YOZ* plane.

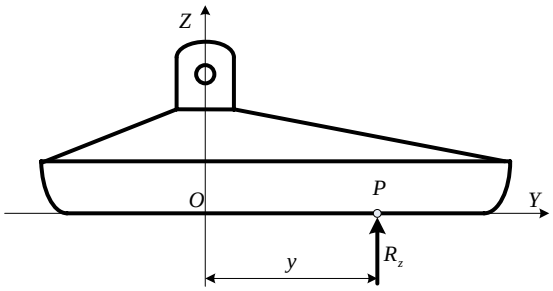


Fig. 5.4: M_{Ax} compensation [Vukobratovic 2004]

The moment M_{Ax} is balanced by shifting the acting point of the force R_z. Logically, the reaction force all the time is within the area covered by the foot. A possible increase of the ankle moment will be compensated for only by changing (shifting) the position of the application of the force R_z and no horizontal components of the moments M_x and M_y will exist. When the ankle moment M_{Ax} is too large, the application point of the force R_z balancing it approaches to the edge of the foot and finally, acts upon it. At this moment the uncompensated part of the horizontal components of the reaction moment M_x and M_y appear. They cause the mechanism to rotate around the edge of the foot (overturning). Therefore, the necessary and sufficient condition of the locomotion mechanism to be in the dynamic equilibrium is that for the point P where ground reaction force is acting $M_x = 0$ and $M_y = 0$. In this case, a natural choice to name this point was Zero-Moment Point or ZMP. Thus, over time different definitions of the ZMP have been presented by different researchers but the most appropriate seem to be following two interpretations of the ZMP:

Interpretation 1 [Hirai 98], [Nakamura 99]: *ZMP is defined as that point on the ground at which the net moment of the inertial forces and the gravity forces has no components along the horizontal axes.*

Interpretation 2 [Arakawa 97]: *P is the point that $M_x = 0$ and $M_y = 0$, M_x and M_y represent the moments around x and y axis generated by reaction force R and reaction torque T, respectively. The point P is defined as the Zero Moment Point (ZMP). When ZMP exists within the domain of the support surface, the contact between the ground and the support leg is stable: $P_{ZMP} = (x_{ZMP}, y_{ZMP}, 0) \in S$, where P_{ZMP} denotes the position of the ZMP. S denotes a domain of the support surface. This condition indicates that no rotation around the edges of the foot occurs.*

Given the mechanism's dynamics, the ZMP position to ensure the dynamical equilibrium of biped mechanism can be computed by stating the static equilibrium equations for the supporting foot (Fig. 5.3):

$$R + F_A + m_s g = 0 \tag{5.1}$$

$$\overrightarrow{OP} \times R + \overrightarrow{OG} \times m_s g + M_A + M_z + \overrightarrow{OA} \times F_A = 0 \tag{5.2}$$

where OP, OG, and OA are radius vectors from the origin of the coordinate system O_{xyz} to the ground reaction force acting point P. The projection of these equations on to a horizontal plane is a basis for computing the position of the point P where the ground reaction force R is applied. If the position of point P computed from equation (5.1) and (5.2) is within the support polygon, the system is in dynamic equilibrium and point P can be called the ZMP position. If the computed position of point P is outside the support polygon, this means that the ground reaction force acting point P is actually on the edge of the support polygon (foot) and the mechanism rotation will be initiated by the unbalanced moment $M_x \neq 0$ and $M_y \neq 0$. The intensity of these moments depends on the distance from the support polygon edge to the computed position of the point P. In reality, point P cannot exist outside the support polygon, as

in that case the reaction force R cannot act on the system at all. But if we suppose that the computed position nevertheless lies outside the support polygon, in view of the fact that this position was obtained from the conditions $M_x = 0$ and $M_y = 0$ we can consider it as a fictitious ZMP (FZMP) [Vukobratovic 2004]. However, in most of the works relating to biped robot walking stability authors do not make special reference to the meaning of the ZMP. Thus, every resulting position (inside or outside of the support polygon) of the ground reaction force is called ZMP position. In this work we also will call this point ZMP, taking into account that every time it lies outside the support polygon it is fictitious.

Alter the ZMP is clarified for the part below the ankle (foot) of the biped mechanism, it necessary to consider how it can be interpreted for the entire robot.

In addition to the forces which appeared at the contact point of the biped mechanism with the ground (reaction force and the friction), there are another two principal forces acting on the entire robot's body to take into account. They are gravity and inertial forces.

The human body can be considered as a chain of rigid links (hands, feet, body, etc.), joined by articulations with relative movements between them. Each link is subject to gravity and inertial forces. The best simplification that we can make is to consider that these forces act on the unique point – the Center of Gravity (COG) (Figure 5.5).

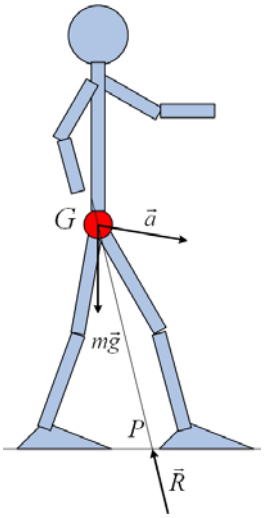


Fig. 5.5: Forces acting on a humanoid (sagittal plane)

In the figure 5.5 point G is the Center of Gravity of the humanoid, P is the point where the ground reaction force $\vec{R}$ is applied, $m\vec{g}$ is the gravity force and $\vec{a}$ is the inertial acceleration of the COG. It is important to mention that inertial forces are provoked by acceleration of the links and entire COG. Its analytical expression is: $\vec{F}_{inertia} = m \cdot \vec{a}$.

The entire walking process of the biped machine can be divided into two basic phases depending on the number of feet that are in contact with the ground. They are:

- The Single Support phase in which the body is supported in a single foot. This phase takes approximately 80% of the total step time.

- The Double Support phase in which the body is supported by both feet. This phase takes approximately 20% of the total step time.

The study of these two phases of motion should be made separately but in both of them the motion can be called stable if no unbalanced overturning moments exist [Goswami 2004]. The motion is called unstable if overturning moments causing the rotation of the biped mechanism about the edge of supporting foot/feet exists. Why do these overturning moments appear? Definitely, considering the entire biped mechanism (Figure 5.6), the instability is provoked by misalignment of the ground reaction force $\vec{R}$ (acting in that case on the foot edge) and resulting of the gravity $m\vec{g}$ and inertial $\vec{F}_{inertia} = m \cdot \vec{a}$ forces (Figure 5.6(a)).

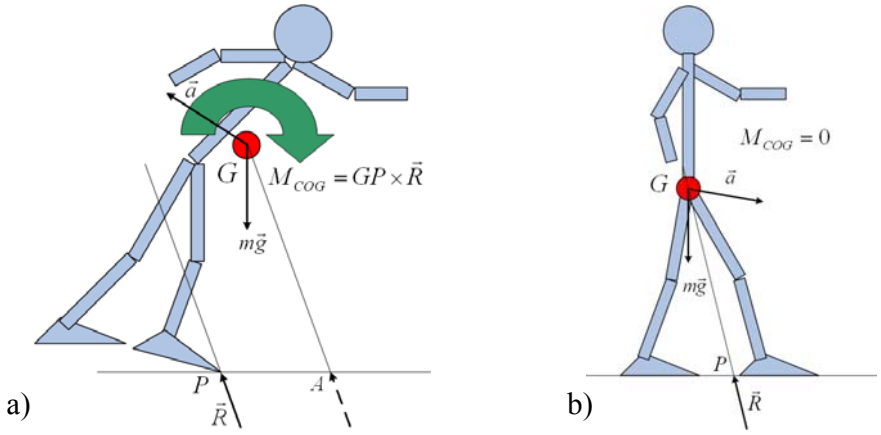


Fig. 5.6: Humanoid robot a) Instable motion b) Stable motion

One of the ways to make certain that forces are in alignment is to ensure that the action line of the resulting ground reaction force $\vec{R}$ passes through the Center of Gravity (Figure 5.6(b)). In that case all moments in the point G are compensated and its total sum is zero ($M_{COG} = 0$). Also, it can be concluded that in the case of a stable dynamically balanced gait ZMP coincides with the Center of Pressure point P [Vukobratovic 2004]. When the gait is not stable, ZMP does not exist (FZMP exists) and the biped mechanism is overturning about its foot/feet edge. Finally, ZMP for the entire humanoid mechanism is a dynamical parameter that can be used for measuring walking stability of the biped locomotion.

The stability theory introduced by Vukobratovic and discussed above for the case of the Single Support phase says that in order to secure stability of the biped mechanism and so as not to cause the mechanism's rotation by the unbalanced moments, the ZMP should be located inside the support polygon. To unify the use of ZMP for both walking phases it is necessary to introduce the concept of a stability region.

In the Double Support phase the stability region is determined as a support polygon of the foot in contact with the ground (striped area in the figure 5.7(a)).

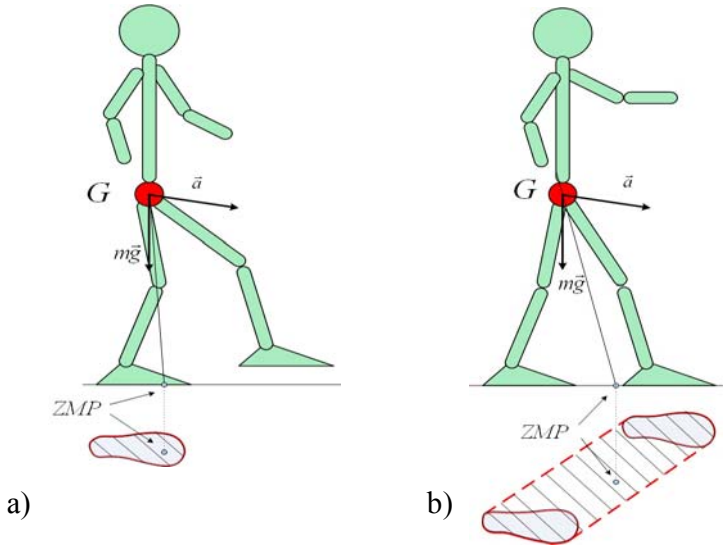


Fig. 5.7: Stability region for a) Single Support Phase b) Double Support Phase

This phase continues for longer in the step cycle and has relatively a small region of the ZMP stability. It makes the Single Support phase more significant and complicated for stability maintenance. In the Double Support phase the stability region is determined as support polygons of both supporting feet plus the surface between them (the striped area in the figure 5.7(b)).

Logically, and in compliance with the ZMP concept, if it is located every time inside the stability region, the walking action of the biped machine is stable. In this context, knowledge of the ZMP position is an essential condition for designing the motion of the biped robot. In a general case the determination of ZMP requires knowledge of all forces and moments acting on the mechanism during its motion and is known as a dynamical approach. The computation is extremely complex because of the high number of DOFs and the wide mobility of the humanoid mechanism. However, there are some methods which allow for modeling of the humanoid robot and designing its motion using dynamical and ZMP constraints. In the following section one of the methods to generate stable humanoid robot motion patterns - the Inverted Pendulum Mode will be presented.

Finally, although it seems not to be especially suitable to study the motion in a static equilibrium, the pseudo-ZMP concept should also be mentioned. By definition the pseudo-ZMP is the projection of the COG on the ground (Figure 5.8).

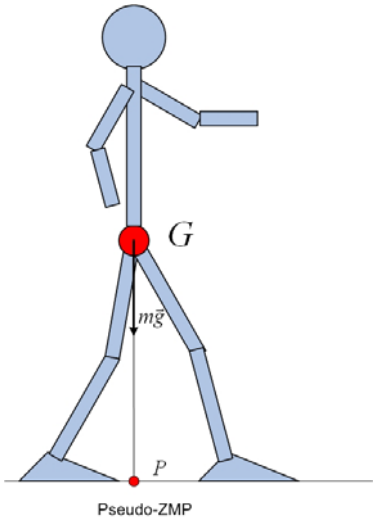


Fig. 5.8: Pseudo-ZMP

The pseudo-ZMP concept can be useful in a case when a humanoid robot moves with a very low speed (so called Static Gate). In this case lateral accelerations are relatively small and can be omitted without creating big errors, thus simplifying computation of the ZMP. By this, we can assume that for the low speed humanoid motion pseudo-ZMP = ZMP. Later in following sections it will be shown that the pseudo-ZMP is the ZMP if we do not take into account the influence of the inertia. Although the pseudo-ZMP does not provide the exact information about the stability of the mechanism, it can be used for the first approximation in control and design of a humanoid robot.

5.3 Biped walking dynamics modeling and patterns generation

5.3.1 Basic approaches

A humanoid robot has a very complex dynamics characterized by its complex mechanical configuration. Figure 5.9 presents a simplified mechanical configuration (only the basic joints) of a typical humanoid robot.

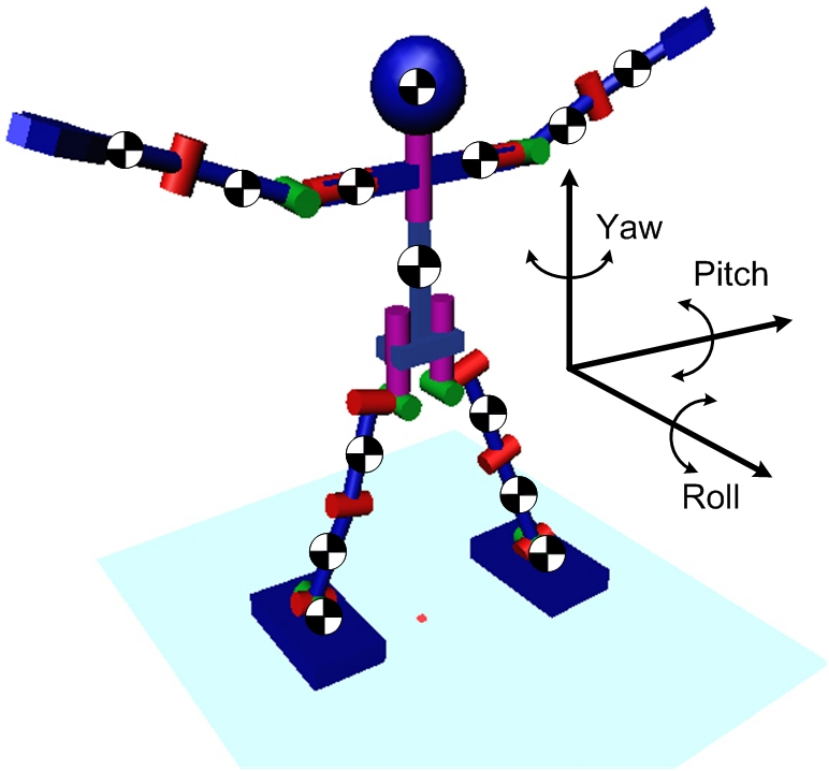


Fig. 5.9: Generalized mechanical structure of a humanoid robot

As humanoid robots have many DOFs it is very difficult to use their dynamics directly to generate a stable biped locomotion, although some research in this field can also be found [Yamaguchi 99], [Huang 2001], [Hirukawa 2007]. These methods require the precise knowledge of robot dynamics including mass, location of the centre of mass and moments of inertia of each link of a robot [Torre 2004]. A walking pattern is generated by solving the contact wrench equations and by applying the resolved momentum control. Therefore, it mainly relies on the accuracy of the models. Moreover, solving contact equations require the execution of complex operations with multidimensional matrixes requiring a lot of computational resources. This leads to the restricted use of these solutions for real-time dynamic walking generation.

Thus, most recent research usually describes the balance and walking locomotion control using different simplified models. These methods use limited knowledge of dynamics, e.g. location of total centre of mass, angular momentum, etc. In these studies, the biped was usually represented by a planar inverted pendulum with the base representing the foot and the ankle joint. For 3D walking pattern generation a three dimensional inverted pendulum is analyzed. This analysis leads to a simple linear dynamics known as Three-Dimensional Linear Inverted Pendulum Mode (3D-LIPM) [Kajita 2001].

Another method models the dynamics of a humanoid robot as a running cart on the table which gives a convenient representation for treating the ZMP. Moreover, this method formalizes the problem of a motion pattern generation as a design of the ZMP tracking servo controller adopting the preview control theory [Kajita 2003]. Preview control tries to change humanoid's

inputs (motion patterns) by simulating the system forward to prevent some constraint perturbations.

Both of these methods for biped locomotion generation are simpler than the direct methods, considering the entire dynamics, and provide good solutions for real-time pattern generation.

However as will be clearly seen later, while the Cart on the Table model provides the relation ZMP-COG that is very useful for the trajectory generation, the Inverted Pendulum Mode provides a COG-ZMP relation that is useful for the stabilization control of these already generated trajectories. Therefore, in this chapter we will consider 3D-LIPM more in details in order to get clear mathematical representation of a humanoid robot for the further design of the stabilization control. Thus, 3D-LIPM was chosen instead the Cart on the Table model (recently it is considered as a more flexible and powerful tool for motion generation) because it can easily explain the humanoid's dynamics in such a way that afterwards will be used in the design of proposed stabilization control.

5.3.2 Inverted Pendulum Mode

When a humanoid robot in its walking cycle is in the Single Support phase, its dynamics can be represented by a single inverted pendulum where the supporting foot is the point where all body mass is concentrated (COG) and a mass less telescopic leg (Figure 5.10) [Kajita 2001]. The pendulum can increase or decrease its length, thus simulating the functioning of the ankle joint. By this model, we assume that ground reaction force is acting at the origin O (the only contact point of the pendulum with the ground). Inertial $m\vec{a}$ and gravity $m\vec{g}$ forces act on the point mass. $\vec{F} = \left[f_x, f_y, f_z\right]^T$ is a total linear force exerted on the mechanism. The position $\vec{p} = \left[x, y, z\right]^T$ of the point mass concentrated in the COG is denoted by the set of state variables $q = (\vartheta_r, \vartheta_p, \vec{r})$.

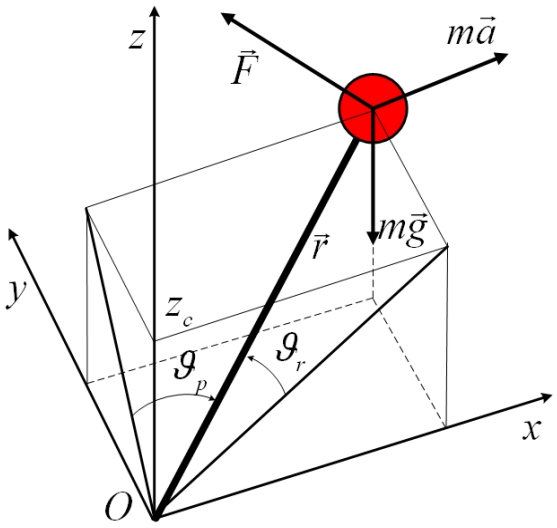


Fig. 5.10: 3D Inverted Pendulum [Kajita 2001]

Applying direct kinematics equations:

$$\begin{cases} x = r \cdot \sin \vartheta_p \\ y = -r \cdot \sin \vartheta_r \\ z = r \cdot D \end{cases} \tag{5.3}$$

where $D = \sqrt{1 - \sin^2 \vartheta_p - \sin^2 \vartheta_r}$.

Applying Newton's formulation of Equation of Motion to presented dynamical system we can obtain:

$$m(\vec{a} + \vec{g}) = \vec{F} \tag{5.4}$$

Let $\tau = (\tau_r, \tau_p, f)$ be the actuator's torques and force associated with state variables $q = (\vartheta_r, \vartheta_p, \vec{r})$ needed to maintain the system in the dynamical equilibrium. The relationship between joint torque τ and the end point force is known as force's direct kinematics and can be written as:

$$\tau = J^T \vec{F} \tag{5.5}$$

Therefore, the inverse kinematics is denoted as:

$$\vec{F} = (J^T)^{-1} \cdot \tau \tag{5.6}$$

Using equation (5.6), the equation (5.4) of motion of the pendulum can be rewritten as:

$$m\vec{a} + m\vec{g} = (J^T)^{-1} \cdot \tau \tag{5.7}$$

Taking into account that $\vec{a} = [\ddot{x}, \ddot{y}, \ddot{z}]^T$ in Cartesian coordinates we have:

$$m\begin{pmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{pmatrix} = (J^T)^{-1} \cdot \begin{pmatrix} \tau_r \\ \tau_p \\ f \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ -mg \end{pmatrix} \tag{5.8}$$

The Jacobian matrix J is computed from (5.3) as:

$$J = \frac{\partial p}{\partial q} = \begin{pmatrix} \frac{\partial x}{\partial \vartheta_r} & \frac{\partial x}{\partial \vartheta_p} & \frac{\partial x}{\partial r} \\ \frac{\partial y}{\partial \vartheta_r} & \frac{\partial y}{\partial \vartheta} & \frac{\partial y}{\partial r} \\ \frac{\partial z}{\partial \vartheta_r} & \frac{\partial y}{\partial \vartheta} & \frac{\partial z}{\partial r} \end{pmatrix} = \begin{pmatrix} 0 & r\cos\vartheta_p & \sin\vartheta_p \\ -r\cos\vartheta_r & 0 & -\sin\vartheta_r \\ \frac{-r\sin\vartheta_r\cos\theta_r}{D} & \frac{-r\sin\vartheta_p\cos\theta_p}{D} & D \end{pmatrix} \tag{5.9}$$

In order to erase the inverse Jacobian from (5.8) it is useful to multiply the matrix J^T from the left:

$$m\begin{pmatrix} 0 & -r\cos\vartheta_r & \frac{-r\sin\vartheta_r\cos\theta_r}{D} \\ r\cos\vartheta_p & 0 & \frac{-r\sin\vartheta_p\cos\theta_p}{D} \\ \sin\vartheta_p & -\sin\vartheta_r & D \end{pmatrix}\begin{pmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{pmatrix} = \begin{pmatrix} \tau_r \\ \tau_p \\ f \end{pmatrix} - mg\begin{pmatrix} \frac{-r\sin\vartheta_r\cos\theta_r}{D} \\ \frac{-r\sin\vartheta_p\cos\theta_p}{D} \\ D \end{pmatrix} \tag{10}$$

Multiplying the first row of equation (5.10) to the coefficient $\frac{D}{\cos\theta_r}$, and the second row to the coefficient $\frac{D}{\cos\theta_p}$ and applying direct kinematics equations (5.3) we get dynamics equations that describe the motion of the 3D inverted pendulum along x and y axis:

$$\begin{cases} m(-z\ddot{y} + y\vartheta_r\ddot{z}) = \frac{D}{\cos\vartheta_r}\tau_r - mgy \\ m(z\ddot{x} - x\ddot{z}) = \frac{D}{\cos\vartheta_r}\tau_r + mgx \end{cases} \tag{5.11}$$

In order to simplify further computations it is important to apply some constraints to limit the motion of the pendulum. These constraints also should be suitable for modeling of a walking of humanoid robot. The first assumption is that the point mass moves along the plane intersecting the z axis in the point z_c and with normal vector (k_x, k_y, z_c):

$$z = k_x x + k_y y + z_c \tag{5.12}$$

For the case of a robot walking on a rugged terrain, the normal vector will not be vertical and the z intersection should be the expected average distance of the centre of the robot's mass from the ground. The second derivative of (5.12) will be:

$$\ddot{z} = k_x\ddot{x} + k_y\ddot{y} \tag{5.13}$$

Substituting these constraints into equation (5.11) and carrying out straightforward calculations we obtain:

$$\begin{cases} \ddot{y} = \dfrac{g}{z_c} y - \dfrac{k_x}{z_c}(x\ddot{y} - \ddot{x}y) - \dfrac{1}{mz_c}\tau_x \\ \ddot{x} = \dfrac{g}{z_c} x + \dfrac{k_y}{z_c}(x\ddot{y} - \ddot{x}y) + \dfrac{1}{mz_c}\tau_y \end{cases} \tag{5.14}$$

where $\tau_x = \dfrac{D}{\cos\vartheta_r}\tau_r$, $\tau_y = \dfrac{D}{\cos\vartheta_p}\tau_p$ are virtual inputs introduced to compensate input nonlinearity at (5.14). τ_x and τ_y are moments in x and y directions respectively. In the case of a walking on the flat terrain, constraints can be presented by a horizontal plane ($k_x = 0, k_y = 0$) intersecting the z axis in the point z_c. Therefore we obtain:

$$\begin{cases} \ddot{y} = \dfrac{g}{z_c} y - \dfrac{1}{mz_c}\tau_x \\ \ddot{x} = \dfrac{g}{z_c} x + \dfrac{1}{mz_c}\tau_y \end{cases} \tag{5.15}$$

In the case of the walking on a slope or stairs where ($k_x \neq 0, k_y \neq 0$) equations (5.15) also can be useful. All we need is another constraint which allows for the simplification of the dynamics. Observing equations (5.14) it is possible to note that it will have the same form as equations (5.15) when the term $(x\ddot{y} - \ddot{x}y) = 0$. Let us multiply the first equation of (5.14) by x and the second by y and subtract one from another:

$$(x\ddot{y} - \ddot{x}y) = \frac{gxy}{z_c} - \frac{k_x x}{z_c}(x\ddot{y} - \ddot{x}y) - \frac{x}{mz_c}\tau_x - \frac{gxy}{z_c} - \frac{k_y y}{z_c}(x\ddot{y} - \ddot{x}y) - \frac{y}{mz_c}\tau_y \tag{5.16}$$

and finally, we get:

$$(x\ddot{y} - \ddot{x}y) = -\frac{(x\tau_x + y\tau_y)}{mz} \tag{5.17}$$

Therefore, for the inclined motion plain we will have the same dynamics as in (5.15) only introducing new constraints about the inputs:

$$(x\tau_x + y\tau_y) = 0 \tag{5.18}$$

To conclude, we can note that equations (5.15) are independent linear equations describing the dynamics of the inverted pendulum in case of the horizontal and non-horizontal constraint plane. The two-dimensional version of this model was introduced in 1991 by Kajita and Tani [Kajita

91] and extended to three dimensions in the case of the zero input torque [Hara 97] by Hara and Yokokaw in 1997. In 2001 the dynamics of the humanoid robot was finally formulated as equations (5.15) and titled as Three-Dimensional Linear Inverted Pendulum Mode (3D-LIPM) [Kajita 2001].

It should be noticed, that the original dynamics of the pendulum which can be derived directly from Lagrange-Euler or Newton-Euler formulations of a dynamical model will be not linear. Thus, the 3D-LIPM method has an advantage of a linear dynamics (easier to control) derived without using any approximation.

5.3.3 Relation between COG and ZMP (ZMP equations)

Now, let us consider that the inverted pendulum instead of only one contact point considered in the 3D-LIPM has a contact polygon – the surface in contact with the ground (Figure 5.11).

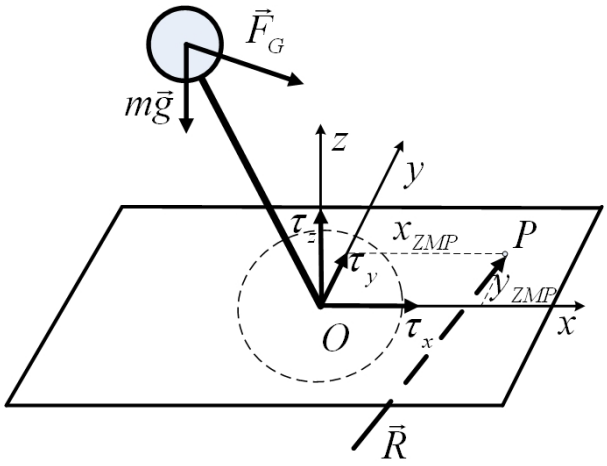


Fig. 5.11: 3D Inverted Pendulum with a contact polygon

As in the previous case, inertial $\vec{F}_G$ and gravity $m\vec{g}$ forces act on the point mass located in the COG of the humanoid robot. The contact of the pendulum with the ground produces its reaction force $\vec{R}$ and reaction moment M_P at point P. For any other point of the support polygon (for example let us take point O) the moment $M_O = [\tau_x, \tau_y, \tau_z]^T$ produced by the ground reaction force $\vec{R}$ will be represented as:

$$M_O = M_P + \overrightarrow{OP} \times \vec{R} \tag{5.19}$$

If we consider that point P is the ZMP of the system, then from the interpretation of the ZMP presented above $M_P = 0$. In this case we can note vector $\overrightarrow{OP} = [x_{ZMP}, y_{ZMP}, z_{ZMP}]^T$ and from (5.19) we will get following equation:

$$\begin{pmatrix} \tau_x \\ \tau_y \\ \tau_z \end{pmatrix} = \begin{pmatrix} x_{ZMP} \\ y_{ZMP} \\ z_{ZMP} \end{pmatrix} \times \vec{R} \tag{5.20}$$

From the other side, applying Newton's law of mechanics to system in the figure 5.11 we have:

$$m\vec{a}_G = \vec{R} - m\vec{g} \tag{5.21}$$

there $\vec{a}_G = [\ddot{x}, \ddot{y}, \ddot{z}]^T$ - is the acceleration of the COG. From (5.21) we can obtain:

$$\vec{R} = m\begin{pmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} + g \end{pmatrix} \tag{5.22}$$

Substituting equation (5.22) into the equation of balance of moments (5.20) we obtain:

$$\begin{pmatrix} \tau_x \\ \tau_y \\ \tau_z \end{pmatrix} = m\begin{pmatrix} x_{ZMP} \\ y_{ZMP} \\ z_{ZMP} \end{pmatrix} \times \begin{pmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} + g \end{pmatrix} \tag{5.23}$$

After carrying out a cross product and taking into account that $z_{ZMP} = 0$ because the ZMP lies into the ground plain we have:

$$\begin{pmatrix} \tau_x \\ \tau_y \\ \tau_z \end{pmatrix} = m\begin{pmatrix} y_{ZMP}(\ddot{z} + g) \\ -x_{ZMP}(\ddot{z} + g) \\ x_{ZMP}\ddot{y} - y_{ZMP}\ddot{x} \end{pmatrix} \tag{5.24}$$

From the equation (5.24) we can state the ZMP position of the mechanism:

$$x_{ZMP} = -\frac{\tau_y}{m(\ddot{z} + g)} \tag{5.25}$$

$$y_{ZMP} = \frac{\tau_x}{m(\ddot{z} + g)} \tag{5.26}$$

If we suppose that the COG always remains within the horizontal plain intersecting the z axis in the point z_c (one of the constraints of the 3D-LIPM model), then the vertical component of the COG acceleration $\ddot{z} = 0$. Then, finally, equations (5.25) and (5.26) take the form:

$$x_{ZMP} = -\frac{\tau_y}{mg} \tag{5.27}$$

$$y_{ZMP} = \frac{\tau_x}{mg} \tag{5.28}$$

From equations (5.27) and (5.28) we can observe that moments τ_x and τ_y in x and y directions respectively affect the ZMP position of the humanoid mechanism. They can cause disequilibrium of the humanoid robot posture and therefore, should always be taken into account. Also, it may be noted that the possible moment in z is not included into the ZMP equation. In the real system it will create the rotation with respect to this axis but without creating any instability.

It is interesting to note that if in equation (5.15) we assume that lateral accelerations of the COG $\ddot{x} = 0$ and $\ddot{y} = 0$ we can get the same equations (5.27) and (5.28) for the ZMP. This proves that in the moment of a static equilibrium (there are no lateral accelerations acting on the pendulum) the ZMP coincides with the pseudo-ZMP (projection of the COG on the ground) mentioned above.

Substituting ZMP equations (5.27) and (5.28) into the equations of COG motion (5.15), finally we get:

$$x_{ZMP} = x - \frac{z_c}{g}\ddot{x} \tag{5.29}$$

$$y_{ZMP} = y - \frac{z_c}{g}\ddot{y} \tag{5.30}$$

Equations (5.29) and (5.30) are called *ZMP equations*. They are the main conditions for generating stable walking patterns and denote the relation between the centre of gravity of inverted pendulum, modeling a humanoid robot, and the ZMP.

5.3.4 Motion patterns generation

Equations (5.29) and (5.30) are second order differential equations. In order to obtain their solution it is necessary to introduce initial conditions (COG position and velocity) by also applying the constraints mentioned above (COG motion is in a horizontal plane intersecting the z axis in the point z_c). After choosing the desired ZMP trajectory form as $x_{ZMP}(t)$ and $y_{ZMP}(t)$ functions, ZMP equations can be solved. Its solutions produce dynamically stable motion patterns and COG trajectory for a humanoid robot walking.

As an example, one of the possible solutions to simplify the computation of the COG trajectory is to assume that for the stable motion the ground reaction force passes through it (Figure 5.12). By this, the point of origin of the coordinates system will always be located at the ZMP.

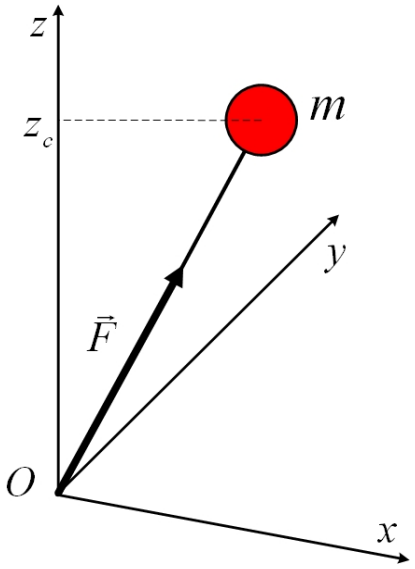


Fig. 5.12: 3D-LIPM with the ZMP in the origin point

Therefore, equation (5.29) and (5.30) can be transformed into:

$$\ddot{x} = \frac{g}{z_c} x \tag{5.31}$$

$$\ddot{y} = \frac{g}{z_c} y \tag{5.32}$$

The solution of second order differential equations (5.31) and (5.32) can be obtained in the following form:

$$x = x_0 \cosh\frac{t}{T_c} + \dot{x}_0 T_c \sinh\frac{t}{T_c} \tag{5.33}$$

$$y = y_0 \cosh\frac{t}{T_c} + \dot{y}_0 T_c \sinh\frac{t}{T_c} \tag{5.34}$$

where $T_c = \sqrt{\frac{z_c}{g}}$, (x_0, y_0) and $(\dot{x}_0, \dot{y}_0)$ are initial COG position and velocity. Figure 5.13 shows an example of two different COG trajectories applying different initial conditions.

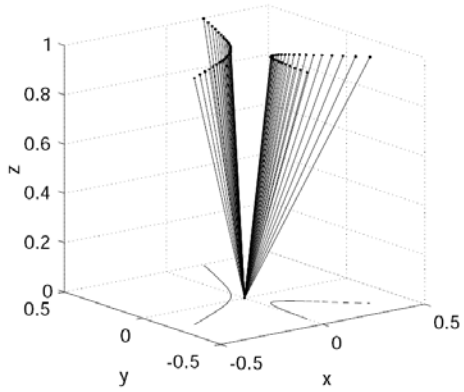


Fig. 5.13: COG trajectories with inicial conditions of the 3D-LIPM [Kajita 2001]

Different papers published by Tokyo University and the National Institute of Advanced Industrial Science and Technology (AIST) of Japan concerning the Inverted Pendulum model are, logically, concentrated on the Simple Support phase because, as was discussed above, it is the more critical part of the walking cycle. In order to generate continuous trajectory for the Double Support phase different hypotheses are introduced, like for example, the constant humanoid robot walking speed. Figure 5.14 presents the motion pattern (position on the upper graph and velocity on the bottom graph) computed using 3D-LIPM with the constant velocity in the Double Support phase [Kajita 2001].

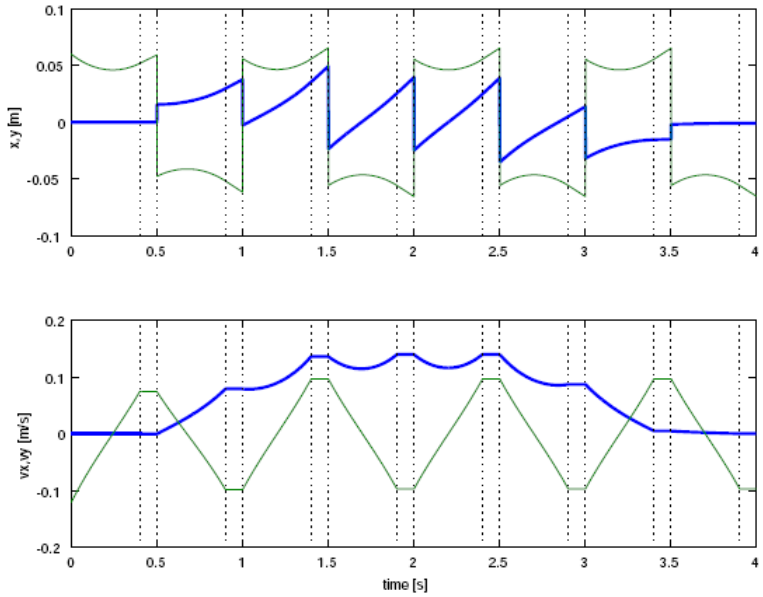


Fig. 5.14: Example of the motion pattern generated using 3D-LIPM [Kajita 2001]

The blue line in figure 5.14 represents position and velocity of the COG in the x direction, and the green line represents the position and velocity in the y direction. This technique for trajectory generation was introduced in the second phase of the HRP project, and implemented in the HRP-2 humanoid robot [Kaneko 2002].

Therefore, before starting the implementation of the stabilizer, the motion pattern should be already generated using the 3D-LIPM or another method. Summing up all that was discussed above, the 3D-LIPM allows us to:

1. Model the human body as an inverted pendulum, i.e. as a mass located in the COG of the robot's body and a mass less telescopic leg joining the COG with the leg.
2. Simplify the dynamics Supposing that the COG moves along a horizontal plane intersecting the z axis at point z_c. However, in this case we have an error to take into account (which will be considered in the next section) because in the real system the COG does not remain in the same horizontal plain.
3. With known ZMP compute the position and acceleration of the humanoid robot's COG.

Finally, as this work does not have as a main objective the study of the problem of humanoid motion generation, no further details will be given on this topic, which has already been well discussed in [Kajita 2001], [Kajita ICRA01]. It should be highlighted that the design of the humanoid stabilizer control which will be presented in following sections was made taking into account that all trajectories are generated using the 3D-LIPM dynamic model, although it can be implemented with any other motion pattern. As was mentioned above, a more powerful approach for generating a motion pattern for stable bipedal locomotion is a Cart on the Table method proposed by Kajita in 2003 [Kajita 2003].

Then, after a COG trajectory has been generated, the inverse kinematics problem for the entire humanoid robot body using different (Denavit-Hartenberg, Lie Logic, etc.) techniques should be solved and the trajectory vectors for each articulation $q_i(t)$ have to be obtained. These trajectories are used as offline calculated motion patterns and denote foot, arms and the entire body trajectory of the robot. Although planned motion patterns satisfy the stability constraints, some errors can cause the humanoid robot to fall over. To reduce the influence of these errors on walking stability, the posture control for online motion patterns modification should be implemented.

5.4 Stabilization Control

5.4.1 The need for stabilization control

Early work on the stability of bipedal robot was done by Vukobratovic [Vukobratovic 70] and followed by Galliday [Galliday 76] and H. Hemami [Hemami 84]. In these studies, the biped robot was usually represented by a planar inverted pendulum with the base representing the foot and the ankle joint. Preview control [Kajita 2003] (a part of cart on the table algorithm considered in the previous section) tries to change humanoid inputs (motion patterns) by simulating the system forward to prevent some constraint perturbations. And finally, researchers working on biomechanics divide human balance control into the hip strategy and the ankle strategy [Kuo 95].

It should be noted that starting from this section, a new and original approach for a humanoid robot walking control proposed in this Ph. D. thesis work will be considered.

Since a humanoid is an electromechanical system, it should have all type of errors (structure flexion, small backlash between motion parts, etc.) proper to this kind of mechanism. Also it will operate in a co-existing environment with humans, an unexpected contact with human or environment (external forces) can be disturbances at any time. Moreover, some imperfections of the dynamical model (which will be discussed later) also bring perturbation into the humanoid's motion. Such disturbances can impede the robot driver controller from following the desired COG and ZMP trajectory derived from equations (5.31), (5.32). In an excessive situation, the humanoid robot can fall over with loss of stability.

Therefore, the Stabilizer is an essential element to provide stable human-like walking of a humanoid robot. The Stabilizer should perform two basic operations:

1. When the humanoid robot walks, it should correct the robot's walking trajectory (motion patterns) in order to provide the secure position at any time of its motion.
2. When the humanoid robot has stopped, it should control its posture.

As was shown in the previous chapter, the humanoid robot's dynamics is governed by the ZMP equations (5.29), (5.30). From these equations it can be concluded that there are two main variables to control:

$$e_{ZMP} = ZMP^d - ZMP^a \tag{5.35}$$

$$e_{COG} = COG^d - COG^a \tag{5.36}$$

where the ZMP error is denoted as e_{ZMP} and the COG position error is denoted as e_{COG}. ZMP^d, COG^d are desired and ZMP^a, COG^a are actual ZMP and COG positions. Equations (5.35) and (5.36) state, that two different controls should be implemented in order to stabilize humanoid robot walking. In figure 5.15 the main structure of the stabilization controller for humanoid robot is presented.

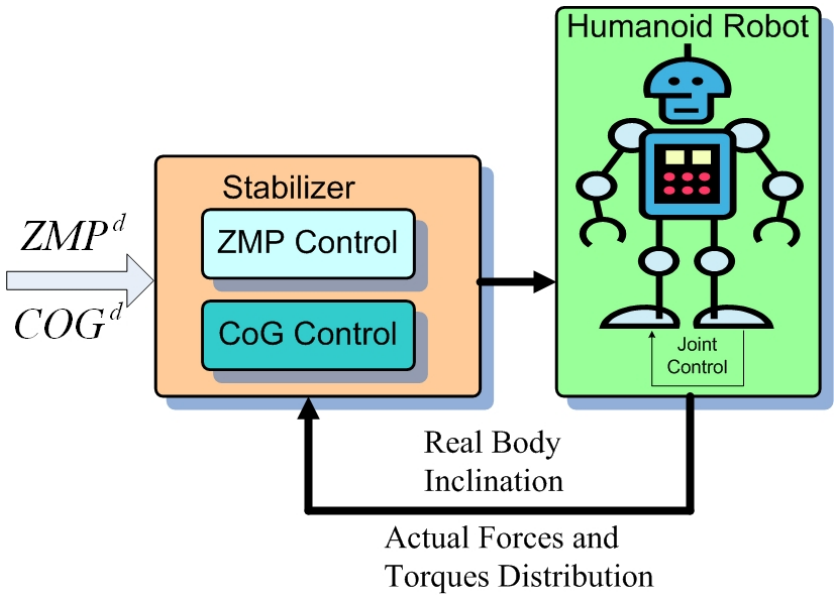


Fig. 5.15: Stabilizer structure

Previously computed from (5.29) and (5.30), the desired COG and ZMP positions generate joint trajectories which are sent for the humanoid motion. While the motion patterns are being executed, the real ZMP and COG position should be measured. The Stabilizer compares it with the ideal ones and generates new corrected motion pattern for the next moment. This control scheme provides a simple and effective way of controlling humanoid walking stability.

However, some questions related to the control implementation can be posed: What is the error indicator for ZMP and COG control and how it can be measured? What kind of action should be implemented on the humanoid robot in order to stabilize it? What is the adequate dynamical model and control law for the ZMP and COG controller? Which actuators of the robot should be implicated into the control algorithm?

The following sections provide a detailed description and try to answer all questions posed above. Also, the design method for the walking (and posture) controller for the humanoid robot will be explained.

5.4.2 ZMP control

When the robot is walking, it is influenced by inertial forces caused by the earth's gravity and the acceleration and deceleration of walking. These combined forces are called the total inertial force. When the robot's foot contacts the ground it is influenced by a reaction from the floor - the ground reaction force. As was shown in previous sections, the intersection of the floor and the axis of the total inertial force have a total inertial force moment of 0, so it is called the Zero Moment Point. Basically, an ideal walking pattern is created from equations (5.30), (5.31) and the robot's joints are moved accordingly. The ZMP of the ideal walking pattern can be called the target ZMP.

When the robot maintains perfect balance, the target ZMP and the real application point of the ground reaction are the same (figure 5.16 (a)). When the robot walks across uneven ground, the axes of the total inertial force which passes through the COG (its intersection with the ground indicates the point where the target ZMP is computed) and the actual ground reaction force $\vec{R}$ are out of alignment, balance is lost and a falling force is generated (figure 5.16(b)). This falling force is proportional to the misalignment of the target ZMP and the real point of the ground reaction force $\vec{R}$ application.

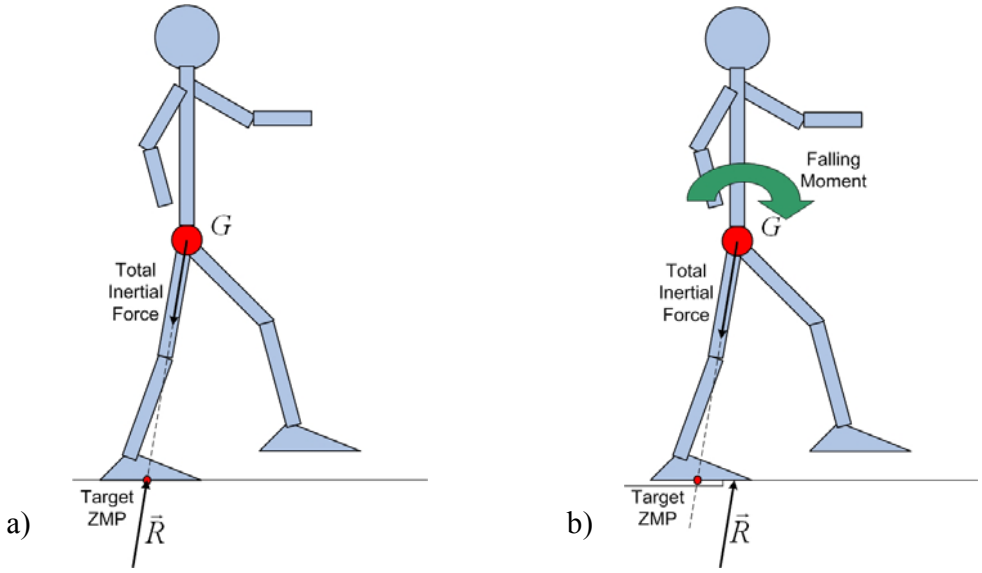


Fig. 5.16: Forces acting on the humanoid robot a) ZMP stable motion b) ZMP unstable motion

In short, the misalignment between the target ZMP and the ground reaction point is the main cause of loss of balance. When a humanoid robot loses its balance and threatens to fall, the ZMP control system should operate to prevent the fall and allow continued walking.

The implementation of the ZMP control should provide an easy and robust method to move the actual ZMP to the point there the target ZMP is located. The easiest way here is to neglect the influence of the part of the humanoid robot above its ankle joint. In figure 5.17 the influence of the upper part of the humanoid robot is replaced by the force $\vec{F}_A$ and moment M_A acting at the point A there the physical ankle joint connection is realized. As the foot experiences the ground reaction at the point P, the total ground reaction force is $\vec{R}$ and the moment M (as was mentioned in previous sections for stable walking M_x and M_y components of this moment should not exist).

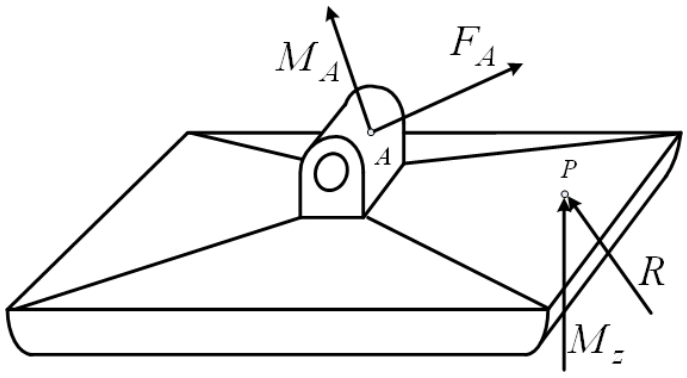


Fig. 5.17: Forces acting on a foot

Considering the dynamical system on the figure 5.17 it can be noticed that in order to shift the point of implementation of the ground reaction force $\vec{R}$ it is necessary to change the moment M_A and the direction of the force $\vec{F}_A$ applied at the point A. This can be done by rotation of the upper part of the mechanism with respect to the ankle joint as is shown in figure 5.18.

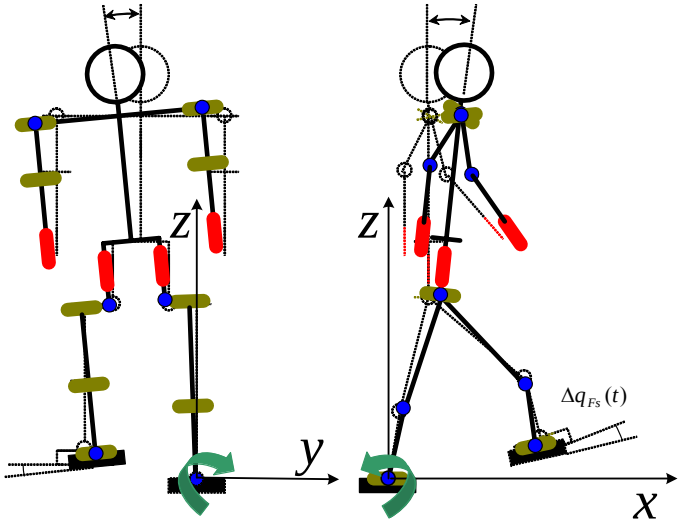


Fig. 5.18: Humanoid robot ZMP error compensation

Although the proposed compensational mechanism allows moving actual ZMP into the point where the target ZMP is located, some problems appear. When the upper body is rotated with respect to the ankle joint, the body angle errors in frontal $\Delta q_{Bf}(t)$ and sagittal $\Delta q_{Bs}(t)$ planes (Figure 5.18) appear. These body angle errors in frontal $\Delta q_{Bf}(t)$ and sagittal $\Delta q_{Bs}(t)$ planes cause the appearance of tilting torques and errors in frontal $\Delta q_{Ff}(t)$ and sagittal $\Delta q_{Fs}(t)$ planes during the positioning of the hanging foot. It can overturn the robot or cause great instability and vibrations when the foot is landing. In order to eliminate these errors a simple compensational mechanism allowing for the correction of the position of the landing foot in frontal $\Delta q_{Bf}(t)$ and sagittal $\Delta q_{Bs}(t)$ planes can be implemented. Also, in the following section the mechanism for compensation of body inclination errors in frontal $\Delta q_{Bf}(t)$ and sagittal $\Delta q_{Bs}(t)$ planes will be considered.

5.4.3 Limits of the 3DLIPM and need of Attitude Control.

In order to describe the need for Attitude Control let us discuss some limits of the Inverted Pendulum dynamical model. One of the main constraints of the 3D-LIPM is that the COG moves in a constant horizontal plane. By this, some simplifications ($\sum F_z = 0$; $\sum M_x = 0$; $\sum M_y = 0$) were made. The absence of the vertical force F_z because of the neglecting of $\ddot{z}$ acceleration of the COG due its location into the constant horizontal plain leads to the absence of horizontal moments M_x and M_y generated by this force. Experiments carried out with humans and

humanoid robots [Arbulu 2005] show that in real walking $\ddot{z} \neq 0$. Force sensors located in the foot and accelerometers located into the COG of a walking human can measure small movements in z axis direction. Therefore, the real COG motion is more similar to the trajectory presented in figure 5.19.

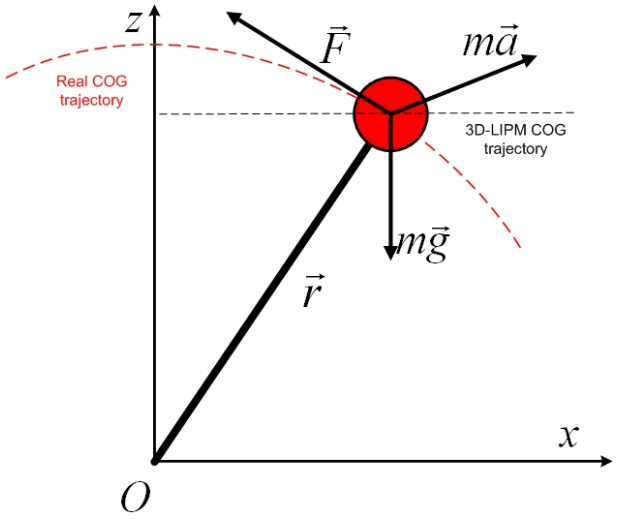


Fig. 5.19: COG trajectory in the sagittal plane

Therefore, considering the existence of the vertical acceleration $\ddot{z} \neq 0$, it is necessary to take into account vertical force F_z and corresponding moments with respect to horizontal axis:

$$\sum M_x = I_y.\alpha_y \; ; \; \sum M_y = I_x.\alpha_x \tag{5.37}$$

where I_x, I_y are moments of inertia and α_x, α_y are angular accelerations for axes x and y respectively. So, the real dynamics includes inertial terms which were excluded from the theoretical inverted pendulum model. Thus, for the multi-link model ZMP equations should take a form:

$$x_{ZMP} = \frac{\sum_{i=1}^{n} m_i(\ddot{z}_i + g)x_i - \sum_{i=1}^{n} m_i \ddot{x}_i z_i - \sum_{i=1}^{n} I_{iy}\alpha_{iy}}{\sum_{i=1}^{n} m_i(\ddot{z}_i + g)} \tag{5.38}$$

$$y_{ZMP} = \frac{\sum_{i=1}^{n} m_i(\ddot{z}_i + g)y_i - \sum_{i=1}^{n} m_i \ddot{y}_i z_i - \sum_{i=1}^{n} I_{ix}\alpha_{ix}}{\sum_{i=1}^{n} m_i(\ddot{z}_i + g)} \tag{5.39}$$

In the case of a single mass concentrated into the COG (inverted pendulum model) we can simplify equations to the form:

$$x_{ZMP} = x - \frac{\ddot{x}z}{(\ddot{z}+g)} - \frac{I_y \alpha_y}{m(\ddot{z}+g)} \tag{5.40}$$

$$y_{ZMP} = y - \frac{\ddot{y}z}{(\ddot{z}+g)} - \frac{I_x \alpha_x}{m(\ddot{z}+g)} \tag{5.41}$$

Comparing (5.40), (5.41) with equations (5.29), (5.30) obtained using 3D-LIPM model it can be noted that they have the same form but additional extra terms because of the real dynamics of the humanoid robot.

The use of equations (5.40), (5.41) to control stability of a humanoid robot seems to be very complicated because it requires continuous measuring of the vertical acceleration $\ddot{z}$ and the real position of the COG at every moment. Therefore, in order to use simplified models with its constraints simplifying the real dynamics, in practice it is necessary to implement a control mechanism capable to compensate moments M_x, M_y acting on the COG. By this, equations (5.40) and (5.41) can be transformed into equations (5.29) and (5.30) without loosing the dynamics of the system.

Also, as was mentioned in the previous section, it is necessary to compensate Body angle errors in frontal $\Delta q_{Bf}(t)$ and sagital $\Delta q_{Bs}(t)$ planes appearing due to mechanical structure flexing and the ZMP control discussed above. Finally, the tilting moment M_{CoG} compounded by M_x and M_y components denotes the robot's upper body dynamics (Figure 5.20) and should be compensated by a control algorithm – the Attitude control.

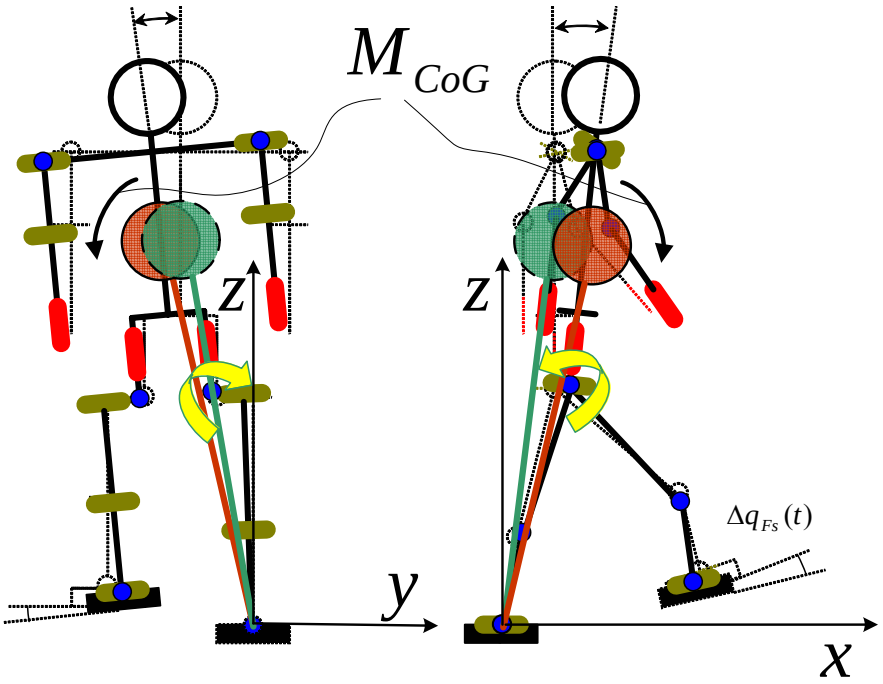


Fig. 5.20: Disturbing moment around COG

The attitude of the robot is usually determined by a combination of data provided by gyroscopes and accelerometers placed in the waist (or upper part of its backbone), where the approximate centre of mass is located. Normally, accelerometers provide correct measurements in a static case, but the information is disturbed by translational accelerations. On the other hand, integration of a gyroscopic angular velocity leads to error due to the high drift rate of the gyroscope. To overcome these problems, it is necessary to process sensory information with the Kalman filtering system and/or High-Low Pass supplementary filters (which will be discussed in the following chapters).

As soon as the sensorial information is obtained, the most effective way to control the humanoid robot's body inclination is to maintain its backbone strictly vertical at every stage of the trajectory. In this case, it is sufficient to control only the hip joints in the frontal and sagittal planes.

5.5 Double Inverted Pendulum

5.5.1 Model design

As was shown in the previous chapters, to maintain the stability of a humanoid robot it is necessary to implement simultaneously both the ZMP and the Attitude controls acting on the ankle and hip joints. The humanoid robot in that case should be modeled as an inverted double pendulum shown in figure 5.21.

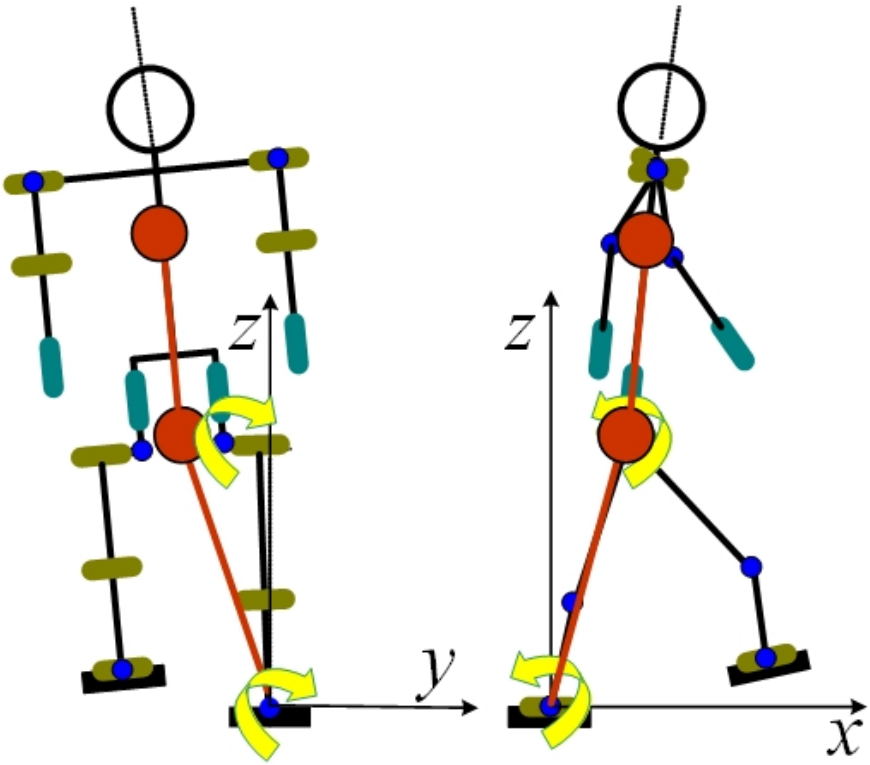


Fig. 5.21: Humanoid robot motion modelling

The humanoid mechanism in this case is divided into two principal parts – the bottom part (biped) and the upper body with its own centers of mass. The motion of the bottom part of the pendulum controls the ZMP and the motion of the upper part controls the body Attitude of the humanoid robot. A mathematical model of the motion of a humanoid robot in the sagittal plane (for the frontal plane dynamical model will be the same) as a double inverted pendulum system is described in figure 5.22.

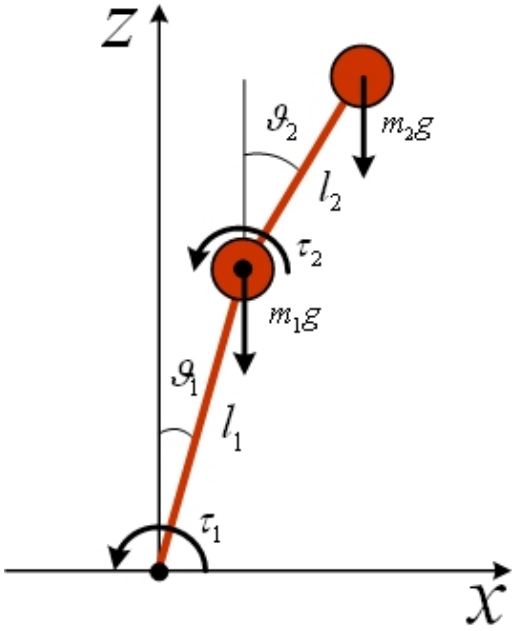


Fig. 5.22: Double inverted pendulum

A double pendulum consists of one pendulum attached to another. Consider a double bob pendulum with masses m_1 and m_2 ($m_1 + m_2 = M$ - the total mass of the humanoid robot) attached by rigid massless wires of lengths l_1 and l_2. Further, let the angles the two wires make with the vertical be denoted ϑ_1 (ankle rotation) and ϑ_2 (hip rotation), as is illustrated in figure 5.22. The position of the centre of mass of the two rods may be written in terms of these angles. Finally, let gravity be given by g. If the origin of the Cartesian coordinate system is assumed to be at the point of contact of the ground and the first pendulum, then the centre of mass is located at:

$$x_1 = l_1 \sin \vartheta_1 \qquad \dot{x}_1 = l_1 \cos \vartheta_1 \dot{\vartheta}_1 \tag{5.42}$$

$$z_1 = l_1 \cos \vartheta_1 \qquad \dot{z}_1 = -l_1 \sin \vartheta_1 \dot{\vartheta}_1 \tag{5.43}$$

$$x_2 = l_1 \sin \vartheta_1 + l_2 \sin \vartheta_2 \qquad \dot{x}_2 = l_1 \cos \vartheta_1 \dot{\vartheta}_1 + l_2 \cos \vartheta_2 \dot{\vartheta}_2 \tag{5.44}$$

$$z_2 = l_1 \cos \vartheta_1 + l_2 \cos \vartheta_2 \qquad \dot{z}_2 = -l_1 \sin \vartheta_1 \dot{\vartheta}_1 - l_2 \sin \vartheta_2 \dot{\vartheta}_2 \tag{5.45}$$

Let us write the equations of the potential and kinetic energies of the dynamic system presented in order to obtain further the dynamical equations governing the double pendulum's motion. The potential energy of the system is given by:

$$V = m_1 g z_1 + m_2 g z_2 \tag{5.46}$$

Substituting the expressions (5.43) and (5.45) into the (5.46) we get:

$$V = (m_1 + m_2) g l_1 \cos \vartheta_1 + m_2 g l_2 \cos \vartheta_2 \tag{5.47}$$

The kinetic energy of the system is given by:

$$T = \frac{1}{2} m_1 v_1^2 + \frac{1}{2} m_2 v_2^2 \tag{5.48}$$

where v_1 and v_2 are the speed of the centers of mass of both parts of the inverted pendulum. As $v_1^2 = \dot{x}_1^2 + z_1^2$ and $v_2^2 = \dot{x}_2^2 + z_2^2$ then using equations (5.42) - (5.44) the kinetic energy of the pendulum takes a form:

$$T = \frac{1}{2} m_1 l_1^2 \dot{\vartheta}_1^2 + \frac{1}{2} m_2 \left[l_1^2 \dot{\vartheta}_1^2 + l_2^2 \dot{\vartheta}_2^2 + 2 l_1 l_2 \dot{\vartheta}_1 \dot{\vartheta}_2 \cos(\vartheta_1 - \vartheta_2) \right] \tag{5.49}$$

The Lagrangian of a system is defined as:

$$L \equiv T - V \tag{5.50}$$

where T and V are expressions for the kinetic and potential energies respectively. Inserting equations (5.47) and (5.49) into equation (5.50) (and simplifying), the Lagrangian becomes:

$$\begin{aligned} L = \frac{1}{2}(m_1 + m_2) l_1^2 \dot{\vartheta}_1^2 + \frac{1}{2} m_2 l_2^2 \dot{\vartheta}_2^2 + m_2 l_1 l_2 \dot{\vartheta}_1 \dot{\vartheta}_2 \cos(\vartheta_1 - \vartheta_2) - \\ - (m_1 + m_2) g l_1 \cos \vartheta_1 - m_2 g l_2 \cos \vartheta_2 \end{aligned} \tag{5.51}$$

When the Lagrangian of a system is known, then the equations of motion of the system may be obtained by a direct substitution of the expression for the Lagrangian into the Euler-Lagrange equation:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\vartheta}_i} \right) - \frac{\partial L}{\partial \vartheta_i} = \tau_i \tag{5.52}$$

Where ϑ_i are the generalized coordinates of the system (in this case they are ϑ_1 and ϑ_2) and τ_i are forces and torques vectors applied to the ϑ_i. Therefore, for ϑ_1:

$$\frac{\partial L}{\partial \dot{\vartheta}_1} = m_1 l_1^2 \dot{\vartheta}_1 + m_2 l_2^2 \dot{\vartheta}_1 + m_2 l_1 l_2 \dot{\vartheta}_2 \cos(\vartheta_1 - \vartheta_2) \tag{5.53}$$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\vartheta}_1}\right) = (m_1 + m_2) l_1^2 \ddot{\vartheta}_1 + m_2 l_1 l_2 \ddot{\vartheta}_2 \cos(\vartheta_1 - \vartheta_2) - \mathrm{m}_2 l_1 l_2 \dot{\vartheta}_2 \sin(\vartheta_1 - \vartheta_2)(\dot{\vartheta}_1 - \dot{\vartheta}_2) \tag{5.54}$$

$$\frac{\partial L}{\partial \vartheta_1} = l_1 g (m_1 + m_2) \sin \vartheta_1 - m_2 l_1 l_2 \dot{\vartheta}_1 \dot{\vartheta}_2 \sin(\vartheta_1 - \vartheta_2) \tag{5.55}$$

Substituting (5.53),(5.54) and (5.55) into Euler-Lagrange equation (5.52) for ϑ_1 and simplifying we get:

$$\begin{aligned}(m_1 + m_2) l_1^2 \ddot{\vartheta}_1 + m_2 l_1 l_2 \ddot{\vartheta}_2 \cos(\vartheta_1 - \vartheta_2) + \\ + \mathrm{m}_2 l_1 l_2 \dot{\vartheta}_2^2 \sin(\vartheta_1 - \vartheta_2) - l_1 g (\mathrm{m}_1 + \mathrm{m}_2) \sin \vartheta_1 = \tau_1\end{aligned} \tag{5.56}$$

Similarly for ϑ_2:

$$\frac{\partial L}{\partial \dot{\vartheta}_2} = m_2 l_2^2 \dot{\vartheta}_2 + m_2 l_1 l_2 \dot{\vartheta}_1 \cos(\vartheta_1 - \vartheta_2) \tag{5.57}$$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\vartheta}_2}\right) = m_2 l_2^2 \ddot{\vartheta}_2 + m_2 l_1 l_2 \ddot{\vartheta}_1 \cos(\vartheta_1 - \vartheta_2) - \mathrm{m}_2 l_1 l_2 \dot{\vartheta}_1 \sin(\vartheta_1 - \vartheta_2)(\dot{\vartheta}_1 - \dot{\vartheta}_2) \tag{5.58}$$

$$\frac{\partial L}{\partial \vartheta_2} = m_2 l_1 l_2 \dot{\vartheta}_1 \dot{\vartheta}_2 \sin(\vartheta_1 - \vartheta_2) + l_2 m_2 g \sin \vartheta_2 \tag{5.59}$$

Substituting (5.57),(5.58) and (5.59) into Euler-Lagrange equation (5.52) for ϑ_2 and simplifying we get:

$$m_2 l_2^2 \ddot{\vartheta}_2 + m_2 l_1 l_2 \ddot{\vartheta}_1 \cos(\vartheta_1 - \vartheta_2) - \mathrm{m}_2 l_1 l_2 \dot{\vartheta}_1^2 \sin(\vartheta_1 - \vartheta_2) - l_2 \mathrm{m}_2 \mathrm{g} \sin \vartheta_2 = \tau_2 \tag{5.60}$$

Regarding the above differential equations (5.56) and (5.60) behaviors of an inverted double pendulum can be analyzed. It is clear that the dynamics of the lower and upper parts of the pendulum is coupled and any small changes in the position of either of them have an influence on the position of the other.

After the dynamical model of a double inverted pendulum was obtained it is necessary to design the control strategy. It should be noted that we do not need to consider swinging the pendulum from the stable position ("all links down") to the "all up" position using the motor input torque. This is called swing-up control. Here we only consider controlling the system once it is already in the upright position. One could assume that in this case, the pendulum links are held upright and stationary while the controller is initialized.

The target of the present research work is to develop a simple control strategy allowing the use of traditional controllers such as the linear quadratic regulator (LQR) or a simple PID controller to regulate the inverted pendulum about the upright equilibrium point. As the name may suggest the LQR controller requires a linear system for which it will generate constant gains for full state feedback to make the equilibrium point globally asymptotically stable. However the dynamics of double inverted pendulum systems are inherently nonlinear. This leaves the problem of how to implement a control methodology designed for a linear system on a nonlinear system. The chosen approach was to linearize the equations of motion at the operating point and define a domain of attraction within which the constant gain controller results in local asymptotic stability.

The key point is that the resulting controller should regulate the both parts of the pendulum (attitude and ZMP control of the humanoid robot) only about zero equilibrium (Figure 5.23)

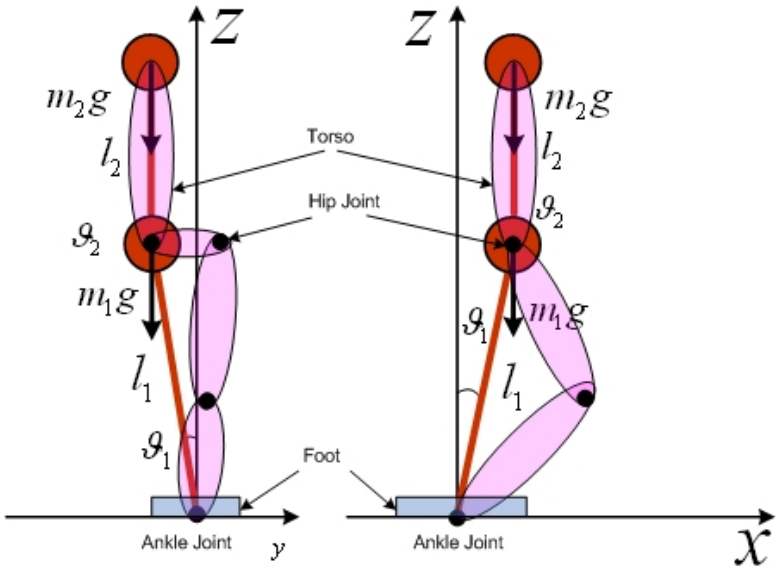


Fig. 5.23: Double inverted pendulum with upper part about zero equilibrium

The state space model was derived as follows. Let us recall the second order nonlinear coupled differential equations (5.56) and (5.60) describing the dynamics of the double inverted pendulum:

$$(m_1 + m_2)l_1^2\ddot{\vartheta}_1 + m_2 l_1 l_2 \ddot{\vartheta}_2 \cos(\vartheta_1 - \vartheta_2) + \\ + m_2 l_1 l_2 \dot{\vartheta}_2^2 \sin(\vartheta_1 - \vartheta_2) - l_1 g(m_1 + m_2)\sin\vartheta_1 = \tau_1 \tag{5.61}$$

$$m_2 l_2^2 \ddot{\vartheta}_2 + m_2 l_1 l_2 \ddot{\vartheta}_1 \cos(\vartheta_1 - \vartheta_2) - m_2 l_1 l_2 \dot{\vartheta}_1^2 \sin(\vartheta_1 - \vartheta_2) - l_2 m_2 g \sin\vartheta_2 = \tau_2 \tag{5.62}$$

For the simplification of a control task let us make a linearization of nonlinear differential equations, taking the approximation that perturbations in the problem are very small and that

terms of second order and higher are negligible. In this case $\cos\vartheta_i = 1$, $\sin\vartheta_i = \vartheta_i$, $\cos(\vartheta_1 - \vartheta_2) = 1$.

It was not defined how small these angles have to be in practice to apply the linearization assumptions. To decide, it is necessary to keep higher order terms and examine their effect on the equations of motion. In this case, it turns out that small ϑ means $\vartheta \le 5^0$.

Applying a small angle approximation we can obtain the following linearization:

$$(m_1 + m_2)l_1^2\ddot{\vartheta}_1 + m_2 l_1 l_2 \ddot{\vartheta}_2 - l_1 g(\mathrm{m}_1 + \mathrm{m}_2)\vartheta_1 = \tau_1 \tag{5.63}$$

$$m_2 l_2^2 \ddot{\vartheta}_2 + m_2 l_1 l_2 \ddot{\vartheta}_1 - l_2 \mathrm{m}_2 \mathrm{g}\vartheta_2 = \tau_2 \tag{5.64}$$

The state representation of the dynamical system in the standard form is:

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \tag{5.65}$$

$$\mathbf{y} = \mathbf{C}\mathbf{x} \tag{5.66}$$

where $\mathbf{x}$ is a state (n - vector), $\mathbf{y}$ is an output (m - vector), $\mathbf{u}$ - control (r - vector), $\mathbf{A}$ - $n \times n$ constant matrix, $\mathbf{B}$ - $n \times r$ constant matrix and $\mathbf{C}$ - $m \times n$ constant matrix.

To obtain the state representation of the double inverted pendulum system let us define state variables x_1, x_2, x_3 and x_4 by:

$$x_1 = \vartheta_1 \tag{5.67}$$

$$x_2 = \dot{\vartheta}_1 \tag{5.68}$$

$$x_3 = \vartheta_2 \tag{5.69}$$

$$x_4 = \dot{\vartheta}_2 \tag{5.70}$$

Note that angle ϑ_1 indicates the rotation of the pendulum about the ankle and ϑ_2 indicates its rotation at the hip. We consider ϑ_1 and ϑ_2 as outputs of the system therefore:

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} \vartheta_1 \\ \vartheta_2 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_3 \end{bmatrix} \tag{5.71}$$

Notice that both ϑ_1 and ϑ_2 are easily measured (using for example joint encoder) quantities. Then, from the definition of the state variables (5.67) – (5.70) and linearized equations of inverted pendulum motion (5.63) and (5.64) we obtain the state space representation of the system:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ \frac{g(m_1+m_2)}{m_1 l_1} & 0 & -\frac{m_2 g}{m_1 l_1} & 0 \\ 0 & 0 & 0 & 1 \\ -\frac{g(m_1+m_2)}{m_1 l_2} & 0 & \frac{g(m_1+m_2)}{m_1 l_2} & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ \frac{1}{m_1 l_1^2} & -\frac{1}{m_1 l_1 l_2} \\ 0 & 0 \\ -\frac{1}{m_1 l_1 l_2} & \frac{(m_1+m_2)}{m_1 m_2 l_2^2} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \quad (5.72)$$

where $\mathbf{u} = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix}$ is a control vector consisting of control torques for both joints of the double inverted pendulum (heap and ankle motor's torques).

The output equation is then:

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \quad (5.73)$$

5.5.2 LQR controller design.

LQR controller design consider an optimal control problem that for given system equations (5.70) and (5.71) determine the matrix $\mathbf{K}$ of the optimal control vector:

$$\mathbf{u}(t) = -\mathbf{K}\mathbf{x}(t) \quad (5.74)$$

so as to minimize the performance index:

$$J = \int_0^\infty (\mathbf{x}^* \mathbf{Q} \mathbf{x} + \mathbf{u}^* \mathbf{R} \mathbf{u}) dt \quad (5.75)$$

where $\mathbf{Q}$ and $\mathbf{R}$ are positive definite Hermitian or real symmetric matrices. It should be noticed that the second term on the right hand of equation (5.75) accounts for the expenditure of the energy of the control signals. The matrices $\mathbf{Q}$ and $\mathbf{R}$ determine the relative importance of the error and expenditure of this energy. The linear control law given by equation (5.74) is the optimal control law. Therefore, if the unknown elements of the matrix $\mathbf{K} = \begin{bmatrix} k_1 & k_2 & k_3 & k_4 \\ k_1' & k_2' & k_3' & k_4' \end{bmatrix}$ are determined so as to minimize the performance index, then $\mathbf{u}(t) = -\mathbf{K}\mathbf{x}(t)$ is optimal for any initial state $\mathbf{x}(0)$. The block diagram showing the optimal controller configuration for the double inverted pendulum system is presented in figure 5.24.

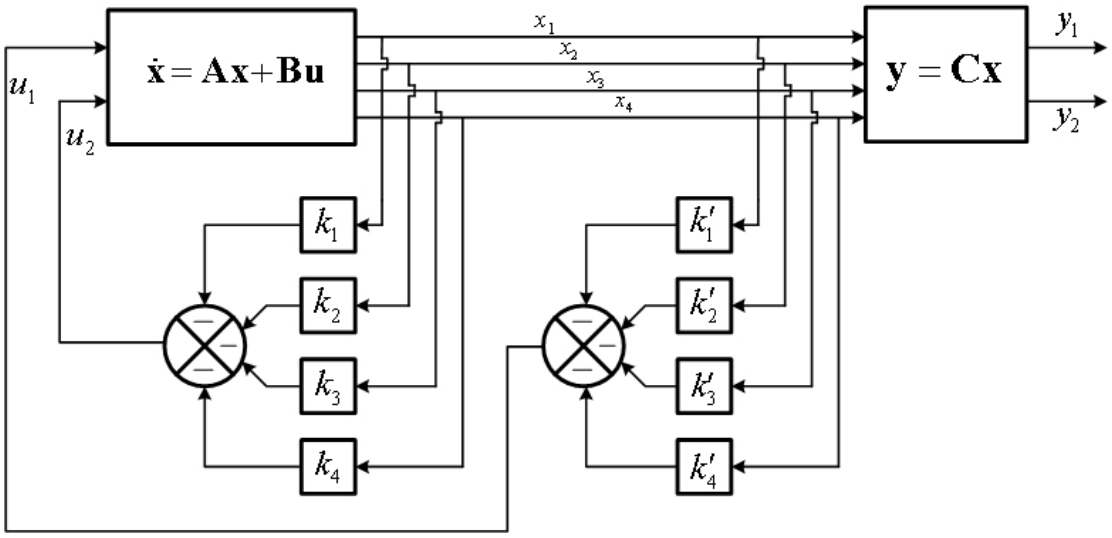


Fig. 5.24: Double inverted pendulum control system

Note that the control system presented in the figure 5.24 is the regulator system. In this case the controller maintains desired angles ϑ_1 and ϑ_2 of the double pendulum close to zero. Thus, the reference input of the control system in figure 5.24 is zero. In the real system hip and ankle joint angles are not zero. These joints, as was shown in previous sections, usually follow the desired trajectory computed for stable biped motion. It means that the real hip and ankle angles should stay as close as possible to its reference value. Thus, a further point of interest for the humanoid robot stabilization control is to have command tracking so that the real humanoid robot joints could be positioned anywhere within the physical bounds. This can be achieved by adding an offset to the desired angle of the ankle and hip joints to make the LQR regulate the double inverted pendulum about zero points.

In order to obtain the controller design for further simulations and discussion, the following mechanical parameters of the inverted pendulum (corresponding to the Rh-1 humanoid robot) were taken: $m_1 = 50\ kg$; $m_2 = 20\ kg$; $l_1 = 1.2\ m$; $l_2 = 0.2\ m$.

For a fast response of the control system the q_{11} element of the $\mathbf{Q}$ matrix in the performance index (5.75) should be taken sufficiently bigger than q_{22}, q_{33}, q_{44} and r_{11}, r_{22} members of the $\mathbf{R}$ matrix. Therefore, we took $\mathbf{Q} = \begin{bmatrix} 100 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ and $\mathbf{R} = \begin{bmatrix} 0.01 & 0 \\ 0 & 0.01 \end{bmatrix}$. After the LQR controller was designed, the control gains matrix $K = \begin{bmatrix} -9.5220 & -2.2725 & 1.2299 & 1.9385 \\ 156.4617 & 9.8653 & -1.9918 & 10.3449 \end{bmatrix}$ was obtained.

Figures 5.25, 5.26 shows simulation results with the designed LQR control system. Position curves in figures 5.25(a) and 26(a) show how the inverted pendulum system returns to its

reference position after different initial perturbations. Control torque curves in the figure 25(b) and 26(b) show the modification of the control torque needed to perform this operation.

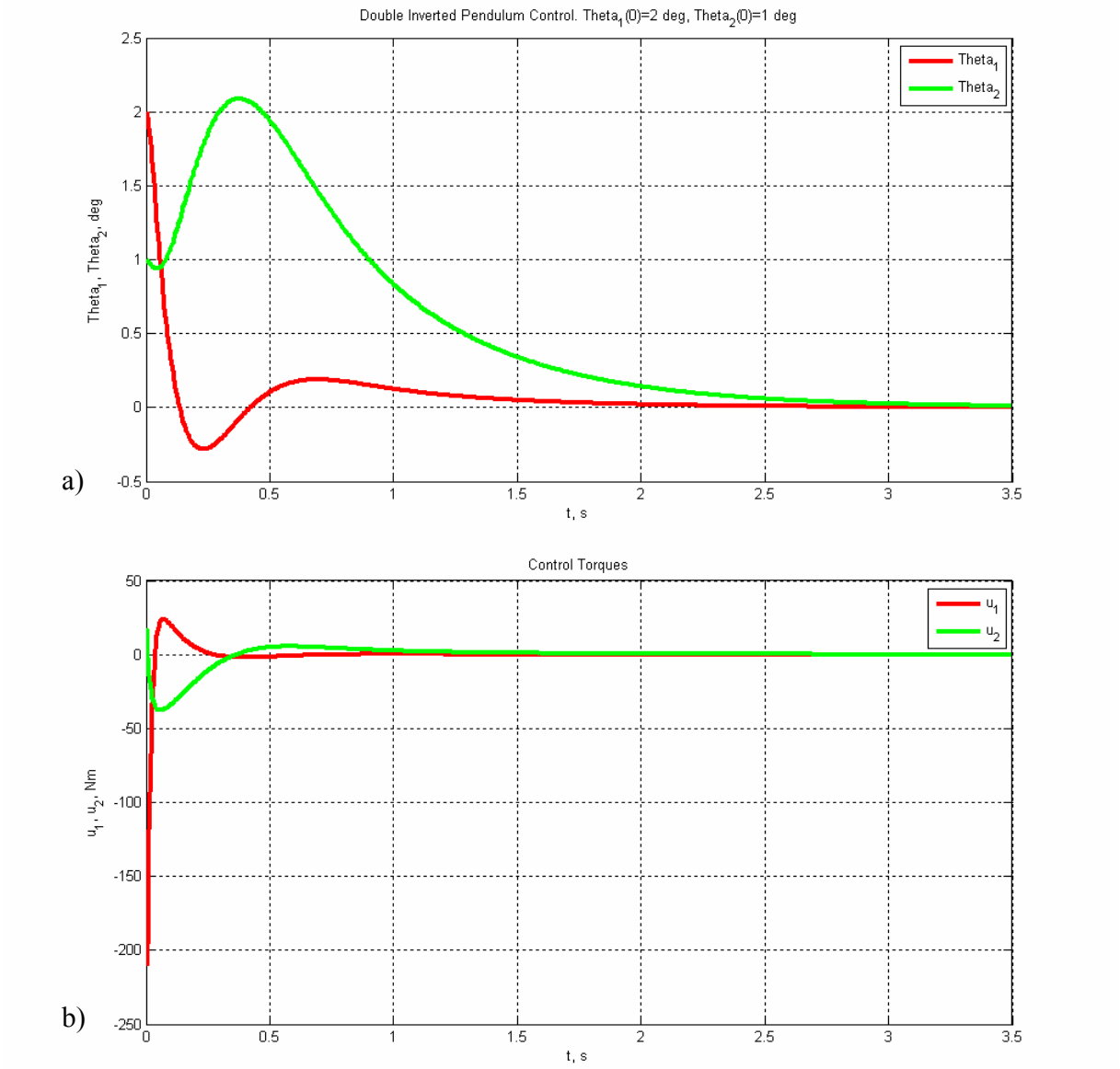


Fig. 5.25: Double inverted pendulum control $\vartheta_1(0) = 2$ deg, $\vartheta_2(0) = 1$ deg a) ϑ_1 and ϑ_2 variation b) Control torques modification

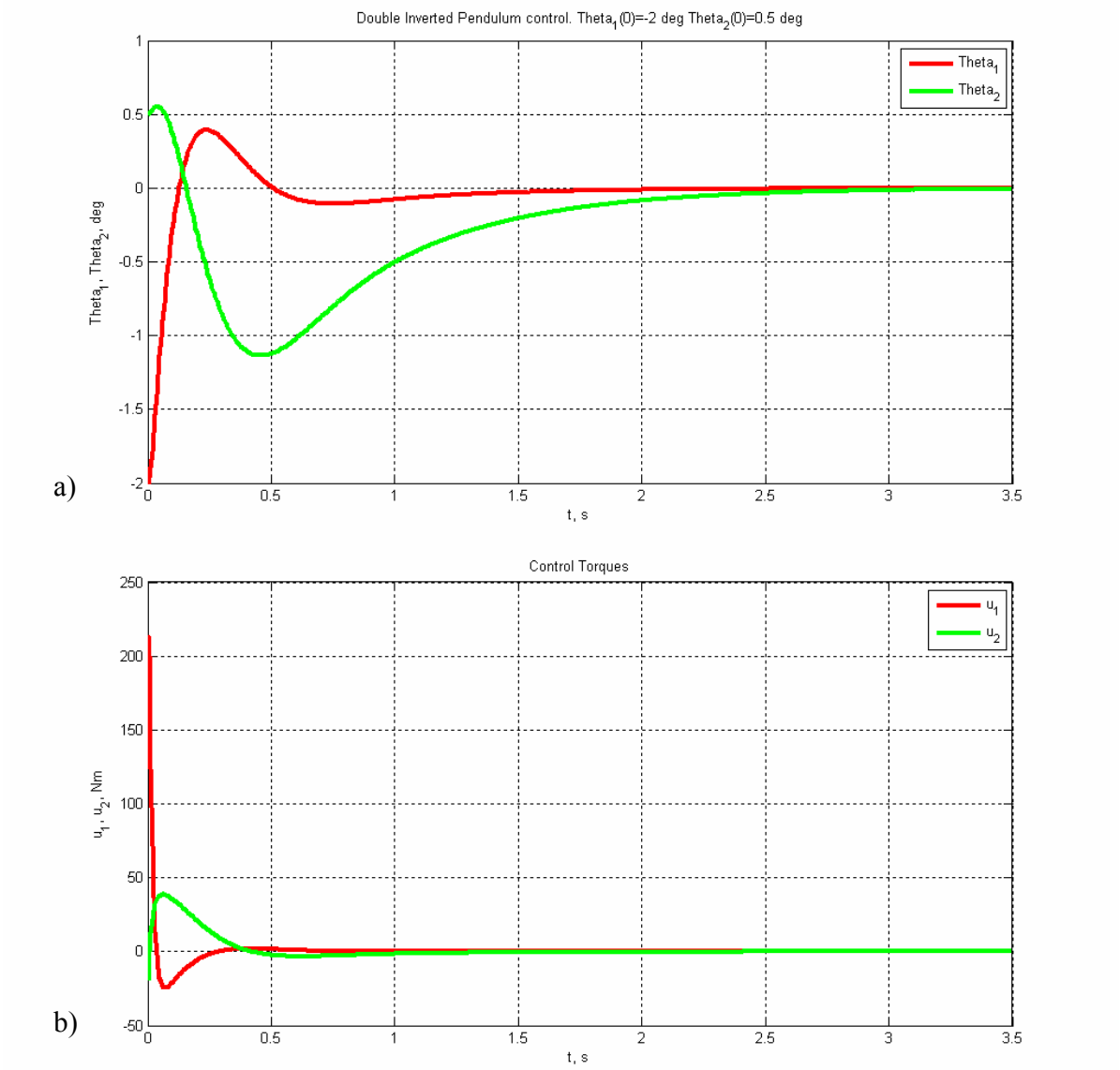


Fig. 5.26: Double inverted pendulum control $\vartheta_1(0) = -2$ deg, $\vartheta_2(0) = 0.5$ deg a) ϑ_1 and ϑ_2 variation b) Control torques modification

It can be observed that response curves are acceptable for controlling the inverted pendulum. The control torques needed to maintain the double inverted pendulum in the desired position are rather small except for in the initial moment when the upswing takes place. Therefore to improve swing up performance the system mass needs to be reduced and the effective control torque increased. Note that the faster response always requires a larger control signal.

5.5.3 Decoupling the control

Although the stabilization method for the humanoid robot based on the double inverted pendulum dynamics presented in the previous section seems to be the most appropriate for controlling the mechanism, in practice, it has many inconveniences. The main reason is that the

double inverted pendulum considers control torques of both (Hip and Ankle) articulations of the humanoid robot involved in the stabilization control. It means there is a need to use the torque controlled actuators to implement this control algorithm. Most contemporary humanoid robots are driven by DC motors with position control where torque control cannot be implemented. Another reason is that the proposed algorithm controls the double inverted pendulum near the desired angular position but does not consider the desired and actual ZMP modifications. And finally, to simplify the stabilization control of the humanoid robot it can be useful to divide it into two independent parts (At the beginning, in the section 5.4 where the need for the stabilizer was proved, the controller was initially proposed as ZMP and Attitude parts). The decoupled approach considers that they can be implemented as totally independent and the influence of one on another in the case of a small initial angular error can be compensated by a controller. In order to prove the appropriateness of this statement let us examine the functioning of already developed control system in diverse initial positions of the pendulum (figure 5.27) that corresponds to different perturbations acting on the stabilizer of the humanoid robot:

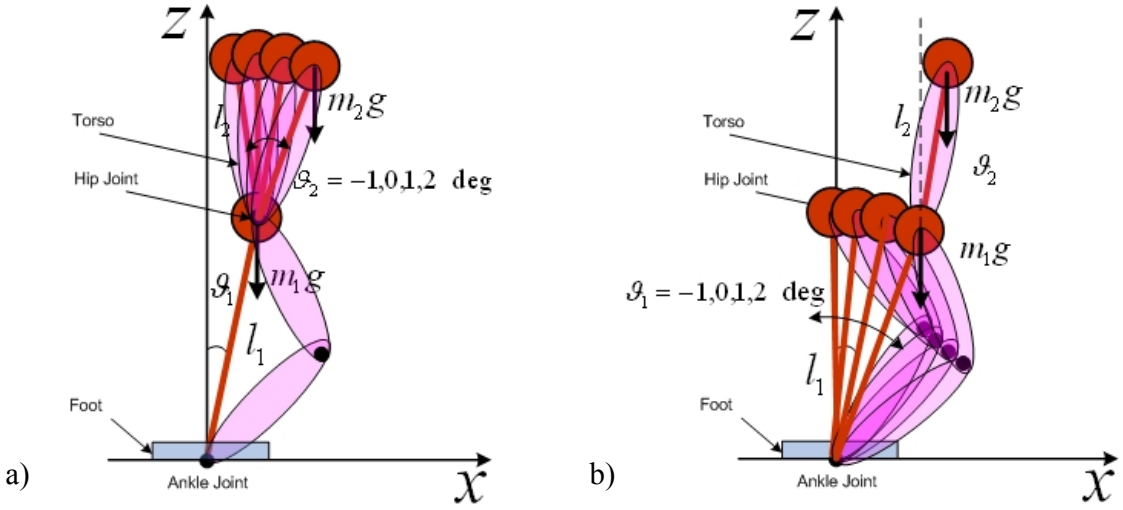


Fig. 5.27: Double Inverted Pendulum a) ϑ_2 variation b) ϑ_1 variation

In the first experiment (figure 5.27(a)), the most critical initial angle ϑ_1 in all experiments was taken as the same while the angle ϑ_2 was starting from different positions in the range from -1° to 2° with the step of 1°, thus, modeling different real perturbations that a humanoid robot suffers while it walks.

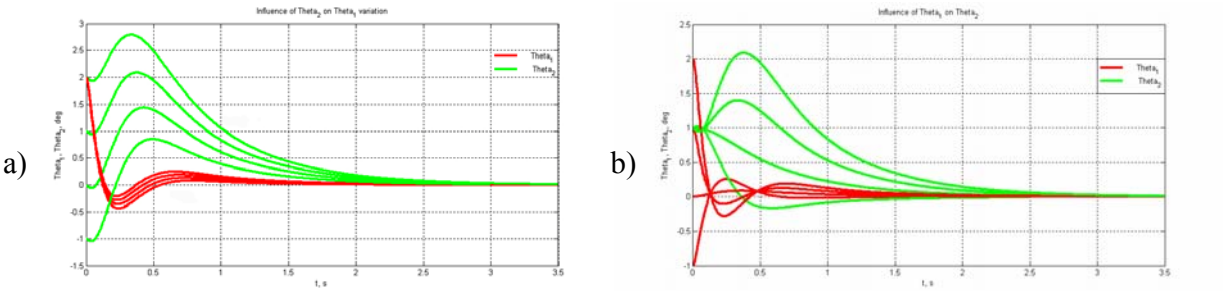


Fig. 5.28: Double Inverted Pendulum control. a)Influence of the ϑ_2 variation on ϑ_1. b) Influence of the ϑ_1 variation on ϑ_2

From figure 5.28(a) it can be observed that large variation of the ϑ_2 does not have much influence on ϑ_1. Its variations stay practically at the same level in every experiment. On the other hand, the variation of ϑ_1 sensed by ϑ_2 (figure 5.28(b)) cannot be neglected, but can be considered as an additional perturbation which should be compensated by its controller before having an effect on the entire dynamics of the system.

In other words, Ankle's variation (ZMP control) affects more the attitude position than the Hip's variation (Attitude control) affects the ZMP. Therefore, Attitude control should be faster in order to compensate the disturbances brought by Ankle's variation.

In order to determine how much faster the control of the upper pendulum (ϑ_2) should be in order to compensate all possible perturbations caused by the motion of the bottom pendulum (ϑ_1)let us consider the dynamics of both of them which can be given in terms of their natural frequency:

$$\omega_n = \sqrt{\frac{g}{l}} \tag{5.76}$$

where l is the pivot length of a pendulum. The relation between natural frequencies of both parts of the dynamic system will be:

$$\frac{\omega_{n2}}{\omega_{n1}} = \sqrt{\frac{l_1}{l_2}} \tag{5.77}$$

Thus, taking as in the previous case $l_1 = 1.2\ m;\ l_2 = 0.2\ m$, the control of the upper pendulum in this case should be at least 2.45 times faster in order to be able to compensate the dynamics of the bottom one.

Therefore, taking into account the constraints stated above, the stabilizer control of the humanoid robot can be considered as a sum of two decoupled components related to Posture control (ϑ_2) and ZMP control (ϑ_1).

Taking into account these considerations, the second order nonlinear coupled differential equations (5.61), (5.62) describing the dynamics of the double inverted pendulum can be transformed into two decoupled dynamical equations:

$$(m_1 + m_2) l_1^2 \ddot{\vartheta}_1 - l_1 g(\mathrm{m}_1 + \mathrm{m}_2)\vartheta_1 = \tau_1 \tag{5.78}$$

$$m_2 l_2^2 \ddot{\vartheta}_2 - l_2 \mathrm{m}_2 \mathrm{g}\vartheta_2 = \tau_2 \tag{5.79}$$

Equations (5.78) and (5.79) are dynamical equations of the single inverted pendulum. Therefore the control task for the stabilization of the humanoid robot finally can be simplified as control of two independent inverted pendulums in frontal and sagittal planes as is shown in the figure 5.29.

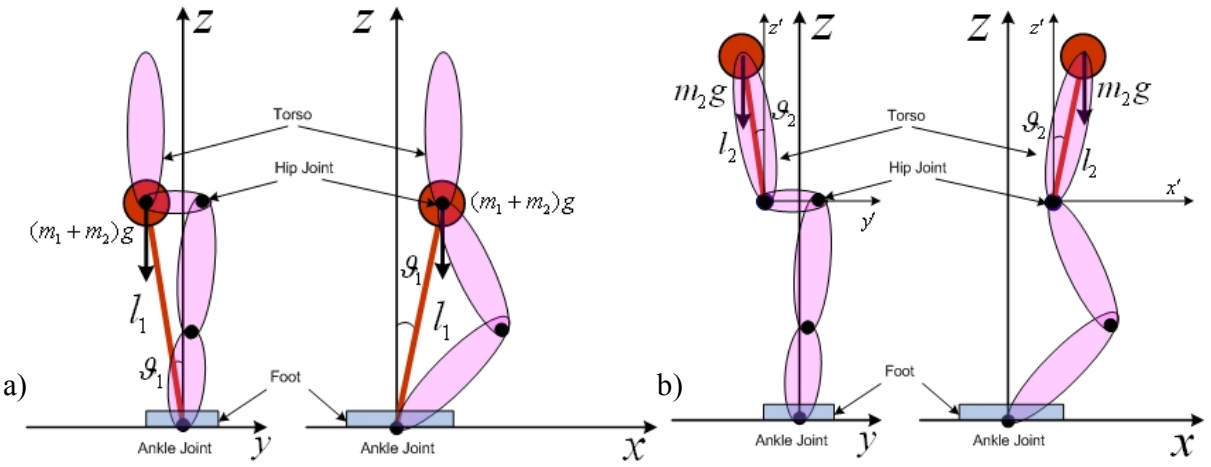


Fig. 5.29: a) ZMP Control b) Attitude control

In this case, ZMP is controlled as an independent inverted pendulum driven by the Ankle joint ϑ_1. Mass $(m_1 + m_2)$ is a total mass of the robot, l_1 is the distance between the Ankle joint and the COG of the humanoid robot. Attitude is controlled as independent inverted pendulum driven by the Hip joint ϑ_2. m_2 is the mass of the upper part (torso) of the humanoid robot and l_2 is the distance from the Hip joint to the center of gravity of the torso.

5.6 Stabilizer

5.6.1 General structure

In the previous section it was shown that the dynamics of a humanoid robot walking can be modeled as a double inverted pendulum. Furthermore, it was demonstrated that the control of a double inverted pendulum in case of a small initial angular error can be decoupled into two independent single pendulum controls. Thus, the structure of the decoupled stabilizer controller consisted of independent ZMP and Attitude controls is shown in Figure 5.30.

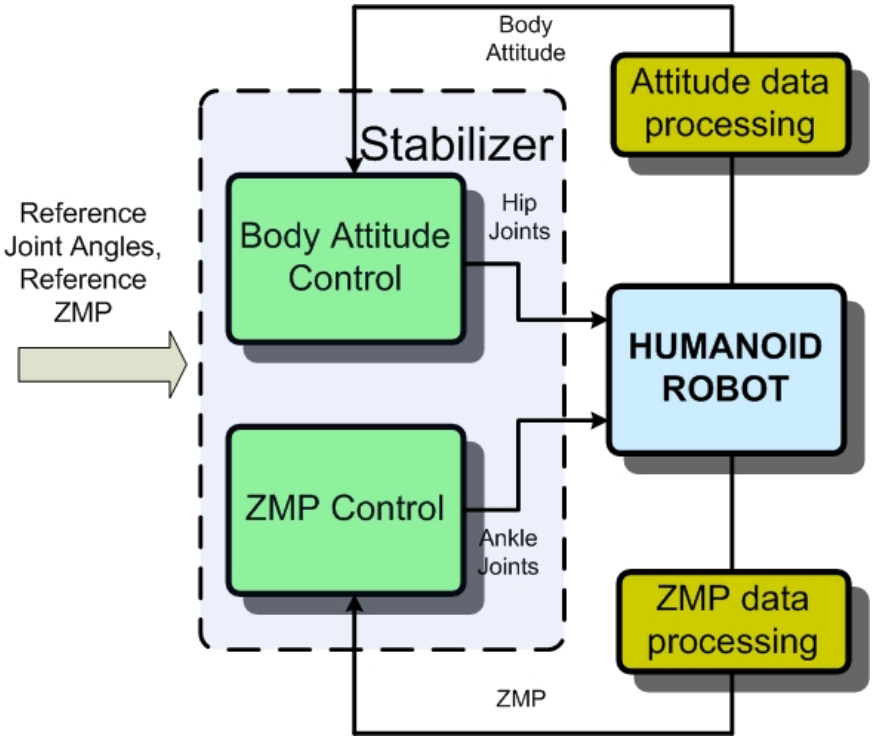


Fig. 5.30: Stabilizer controller structure

The Stabilizer receives real ZMP and Attitude information from sensorial data processing system of the humanoid robot and modifies previously computed reference Hip and Ankle joints angles in order to maintain walking stability. The advantage of this scheme is that there is no any modification of target ZMP or Attitude. All changes are applied to Ankle and Hip joints eliminating the need for inverse kinematics computation. After stabilization of a robot has been achieved, the controller tries to restore the execution of the initial reference pattern. The following sections present detailed design of decoupled ZMP and Attitude controllers as well as the acquisition and processing of the sensorial data needed for these controls.

5.6.2 ZMP controller design.

From the previous section, the model of the bipedal mechanism in a single support phase for ZMP controller design was taken as a single inverted pendulum (fig. 5.29(a)). Its dynamical model in a planar (for example XZ) case is expressed by the equation (5.78). Replacing (m_1+m_2) by a generalized mass m concentrated into the COG of the humanoid robot we obtain:

$$\tau_1 = ml_1^2\ddot{\vartheta}_1 - g\mathrm{m}l_1\vartheta_1 \tag{5.80}$$

where τ_1 is the torque generated by the Ankle joint, ϑ_1 its angular position and l_1 distance between the joint and the COG. The main complexity of this model is the fact that equation (5.80) does not give the possibility of controlling the ZMP by angular position of the Ankle joint. To overcome this problem the inverted pendulum model can be slightly modified. The link of the

pendulum which connects the Ankle joint with the concentrated mass (COG) is generally assumed to be rigid. Nevertheless, in the real humanoid mechanism it is flexible because the leg length is relatively long and the mechanical structure suffers from flexibility and small backlashes. Because of this compliance, the humanoid robot exhibits the characteristics of a lightly damped structure [Kim 2004]. For example, it can be easily observed that in a static case (when the robot had been stopped during the single support phase) when the ankle joint is under position control, the external force (pushing) can easily excite an oscillation. This oscillation exists even when the position error in every joint is zero. In the dynamic case lateral accelerations provoke the same oscillations and errors. It causes errors in the positioning of the landing leg and in the worst scenario the humanoid robot loses its balance and falls over even if the exact motion patterns that satisfy the desired reference ZMP position are known. This phenomenon is prevalent in the fast dynamical gait; therefore it is very important to implement a control mechanism allowing fast ZMP correction considering the stiffness of humanoid robot links. The most suitable model in this case will be a single mass inverted pendulum with compliant joint presented in the figure 5.31 there u denotes the ankle joint reference angle and ϑ denotes the actual inclined angle produced by the compliance of the mechanical structure of the humanoid, K denotes the stiffness of the leg, τ_1 is the torque produced by the motor of the ankle joint to place the inverted pendulum into the desired angular position.

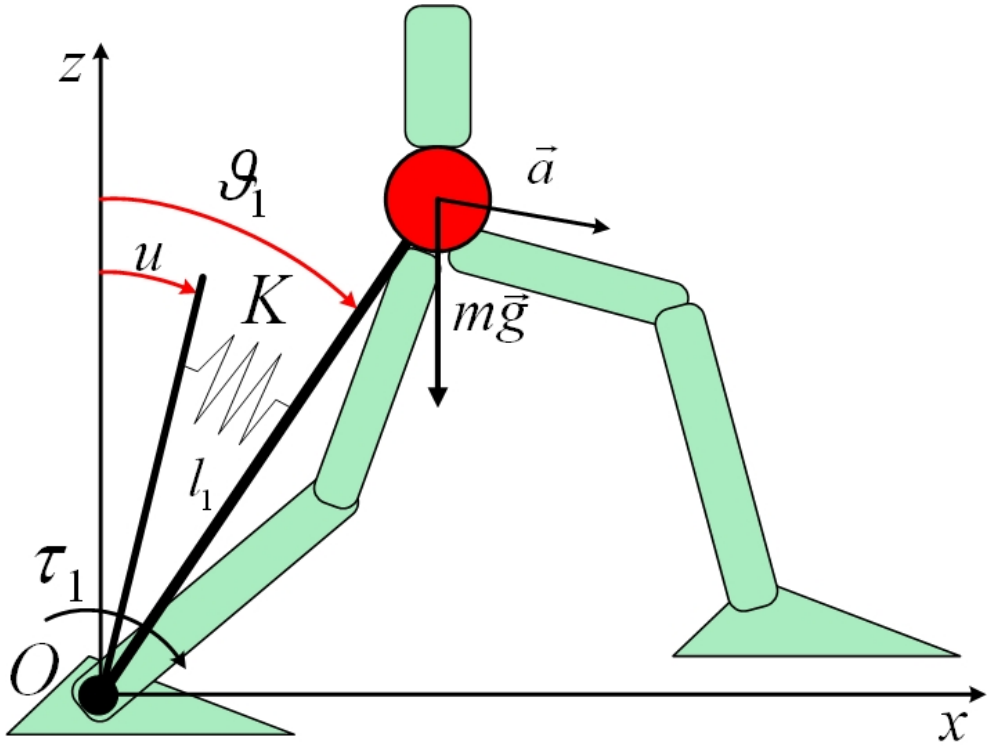


Fig. 5.31: Inverted pendulum with compliant joint

If the damping coefficient C is considered, the torque τ_1 should be expressed as:

$$\tau_1 = K(u - \vartheta) + C(\dot{u} - \dot{\vartheta}) \tag{5.81}$$

However, for reasons of simplicity, the damping coefficient can be neglected because its value is relatively small compared to the stiffness. Thus, the equation describing a spring torque has the form:

$$\tau_1 = K(\vartheta - u) \tag{5.82}$$

From both equations (5.80) and (5.82) the transfer function relating control torque and the command reference angular position can be derived. Taking the Laplace transform of equation (5.80) we obtain:

$$T(s) = ml_1^2 s^2 \vartheta(s) - mgl_1 \vartheta(s) \tag{5.83}$$

The Laplace transform of equation (5.82) is:

$$T(s) = K(\vartheta(s) - u(s)) \tag{5.84}$$

Reflecting $\vartheta(s)$ from the equation (5.84) and placing it into the equation (5.83) and simplifying, we can get the transfer function:

$$\frac{T(s)}{u(s)} = \frac{-Ks^2 + \frac{Kg}{l_1}}{s^2 - \left(\frac{K + mgl_1}{ml_1^2}\right)} \tag{5.85}$$

On the other hand, from the equation (5.27) relating the moment produced by the ground reaction force around y axis with x ZMP direction (the planar XZ case of the inverted pendulum is considered) we can get:

$$\tau_y = -mgx_{ZMP} \tag{5.86}$$

The Laplace transform of the equation (5.86) is:

$$\tau_y(s) = -mgx_{ZMP}(s) \tag{5.87}$$

For the static equilibrium of the system, the moment generated by the motor of ankle joint should compensate the moment produced by the ground reaction force:

$$\tau_1 = \tau_y \tag{5.88}$$

The relation between τ_y and x_{ZMP} is lineal, therefore, placing (5.87) into (5.85) and simplifying we get the following transfer function relating ZMP to Ankle joint position:

$$\frac{x_{ZMP}(s)}{u(s)} = \frac{\frac{K}{mg}s^2 + \frac{K}{ml_1}}{s^2 - \left(\frac{K + mgl_1}{ml_1^2}\right)} \tag{5.89}$$

Equation (5.89) can be rewritten as:

$$\frac{x_{ZMP}(s)}{u(s)} = K_1 \frac{s^2 - (\alpha + \beta)}{s^2 + \alpha} \tag{5.90}$$

where $\alpha = -\left(\frac{K + mgl_1}{ml_1^2}\right), \beta = \frac{K}{ml_1^2}$ and $K_1 = \frac{K}{mg}$.

The unknown parameters of (5.90) can be found by experimental identification. Usually, when the model is complicated it is hard to identify its parameters. However the model described by the equation (5.90) has an advantage in simple and easy parameter identification by experiment. By measuring the oscillation period under position regulation control during single support phase the parameter α can be estimated. By measuring the ZMP increment by ankle joint angle increment at steady state $K_1(\alpha + \beta)/\alpha$ can be obtained. From these two experiments l_1 and K can be estimated. Another way to estimate unknown parameters is by measuring a sinusoidal response for various input frequencies. Also, for the simulation, the approximate value of a stiffness K can be taken from the number of recent relevant researches on the ankles of human subjects [Lakie 2003].

In the equation (5.89) $x_{ZMP}(s)$ is the output and $u(s)$ is the input of the system. It allows for the ZMP of the humanoid robot to be controlled by the position of its ankle joint. In order to design the LQR controller let us transform (5.89) into a space state representation.

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ \frac{K + mgl_1}{ml_1^2} & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u \tag{5.91}$$

The output then is:

$$y = \begin{bmatrix} \frac{K^2}{m^2 l_1^2 g} & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \frac{K}{mg} u \tag{5.92}$$

The state space representation (5.91), (5.92) is a controllable canonical form that is important for the LQR controller design. It is desired to keep the actual ZMP, measured and computed by force-torque sensors located in the foot of the humanoid robot, close to its stable reference position as was discussed in previous sections. As the system is a type 0 plant (without the integrator) it is necessary to insert an integrator in order to design a ZMP servo control system

(type 1). Therefore, we feed the output signal y (which indicates the real ZMP) back to the input and an integrator in the feedforward path as is shown in figure 5.32.

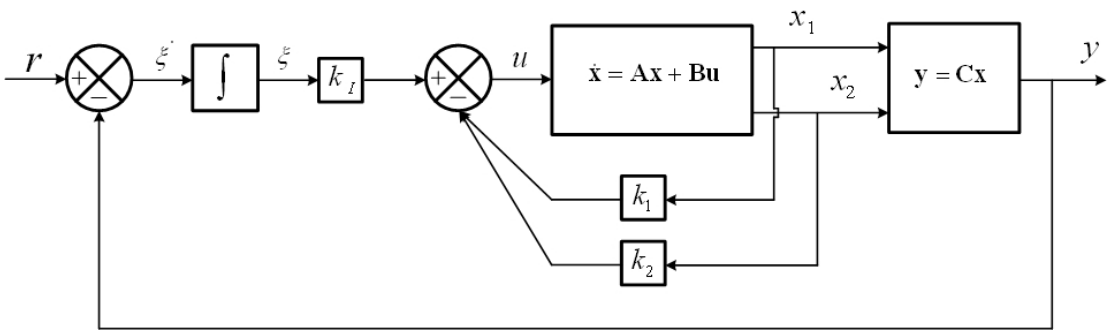


Fig. 5.32: ZMP LQR control system

Thus, referring equations (5.91) and (5.92) and figure 5.32 and considering the actual ZMP position as the output of the system and r as the reference input signal we obtain the equations for the closed loop system as follows:

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}u \tag{5.93}$$

$$y = \mathbf{C}\mathbf{x} + \mathbf{D}u \tag{5.94}$$

$$u = -\mathbf{K}\mathbf{x} + k_I\xi \tag{5.95}$$

$$\dot{\xi} = r - y = r - \mathbf{C}\mathbf{x} \tag{5.96}$$

For the type 1 servo system we will have the state error equation given by:

$$\dot{\mathbf{e}} = \hat{\mathbf{A}}\mathbf{e} + \hat{\mathbf{B}}\mathrm{u}_\mathrm{e} \tag{5.97}$$

where $\hat{\mathbf{A}} = \begin{bmatrix} \mathbf{A} & \mathbf{0} \\ -\mathbf{C} & 0 \end{bmatrix}$, $\hat{\mathbf{B}} = \begin{bmatrix} \mathbf{B} \\ 0 \end{bmatrix}$ and the control signal u_e is given by the equation:

$$u_e = -\hat{\mathbf{K}}\mathbf{e} \tag{5.98}$$

where the state-feedback gain matrix $\hat{\mathbf{K}} = \begin{bmatrix} \mathbf{K} & \vdots & -k_I \end{bmatrix}$.

After the design of the LQR controller some simulations were carried out. Figure 5.33 shows the performance of designed controller $y = x_{ZMP}$ versus t where the input to the system is a reference ZMP step position which equals 0.01 m.

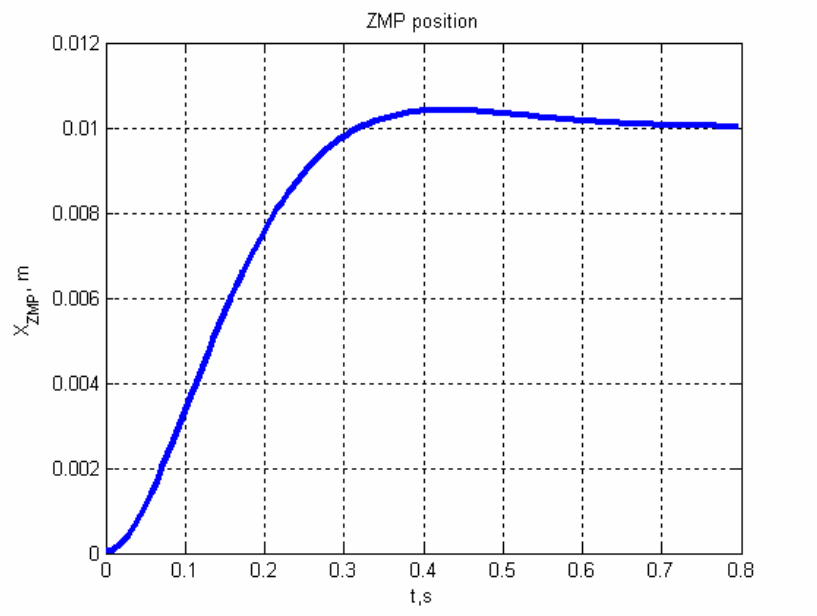


Fig. 5.33: ZMP Control. ZMP modifications

The step response $x_{ZMP}(t)$ shows that the settling time is about 0.6 sec and maximum overshoot is approximately 5% that can be implemented into the ZMP controller of the humanoid robot. The variation of the control ankle joint angle $u(t)$ is shown in the figure 5.34.

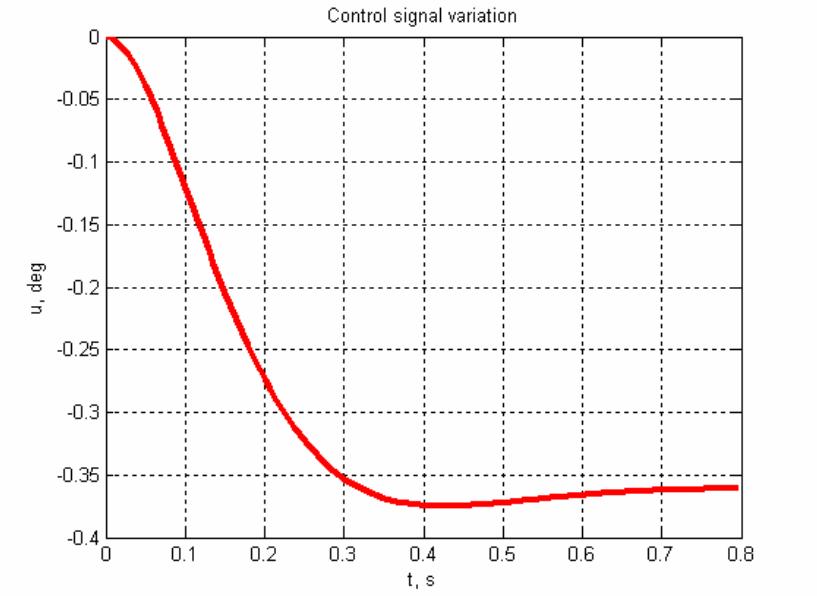


Fig. 5.34: ZMP Control. Control signal modifications

The response curves clearly show that actual $x_{ZMP}(t)$ approaches reference input signal $r = 0.01\ m$ and control ankle joint angle approaches $u = -0.36$ deg needed to maintain this ZMP position. As will be shown in following section, experiments carried out with the OpenHRP simulator prove the range of these signals. Finally, it should be mentioned that ZMP control in the frontal YZ plane can be designed in a same way. The whole humanoid mechanism's motion can be interpreted as a combination of two planar inverted pendulums in frontal and saggittal planes controlled by four independent Ankle joints (2 for the left leg and 2 for the right leg).

5.6.3 Attitude controller design.

From the previous section, the model of the humanoid robot in a single support phase for maintaining its upper part near the zero equilibrium (the trunk in a vertical position) in every stage of the trajectory was taken as a single inverted pendulum. As the trunk is rigid, the simple P controller seems to be sufficient to control it in the vertical position. Nevertheless, in order to model some nonlinearity related to the dynamics of the motor and the inverted pendulum, it is desirable to design a more precise LQR controller. Let us rewrite the dynamical model of the inverted pendulum system expressed by the equation (5.79) in the following form:

$$\tau_2 = m_2 l_2^2 \ddot{\vartheta}_2 - g m_2 l_2 \vartheta_2 \qquad (5.99)$$

where τ_2 is a moment needed to maintain the inverted pendulum in the desired position ϑ_2, l_2 is the distance from the Hip joint to the point where the mass m_2 of the upper body of the humanoid robot is concentrated (Figure 5.35).

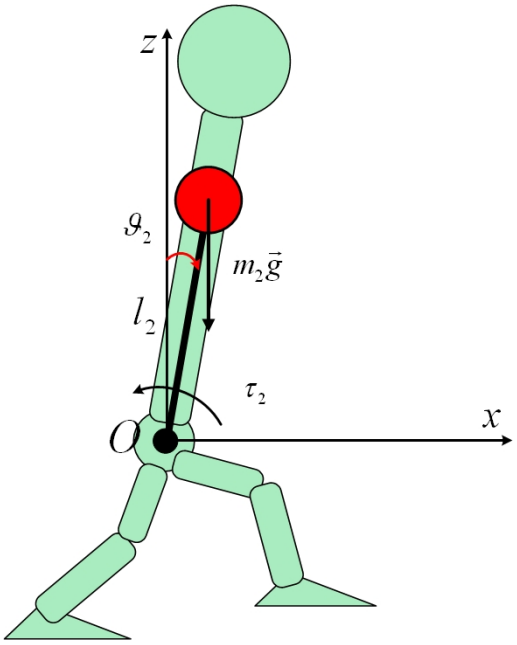


Fig. 5.35: Simple inverted pendulum

On the other hand, a common actuator in the control system of a contemporary humanoid robot is the DC motor. It directly provides rotary motion and, coupled with gears or belt transmission, can provide transitional motion. The electric circuit of the armature and the free body diagram of the rotor are shown in the following figure:

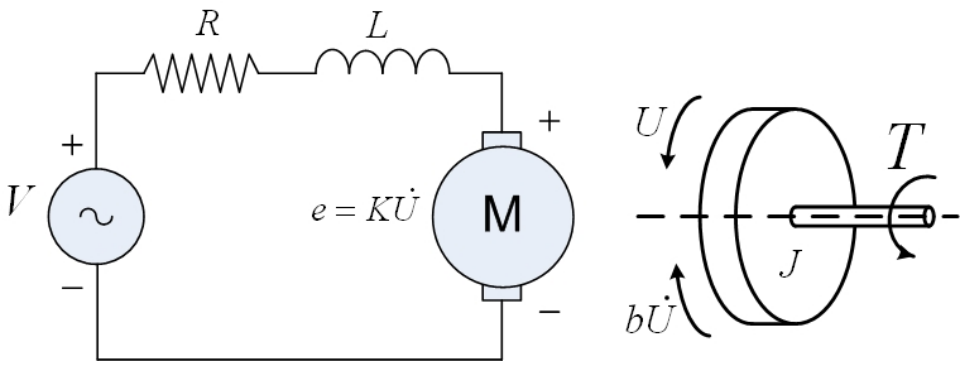


Fig. 5.36: DC motor model

In figure 5.36 J is the moment of inertia of the rotor, b is the damping ratio of the mechanical system, K - electromotive force constant, R – electric resistance, L - electric inductance, V is input source voltage, U is the desired position of the motors shaft. The rotor and shaft are assumed to be rigid. The motor torque T is related to the armature current i by a constant factor K. The back emf e is related to the rotational velocity by the following equations:

$$T = K \cdot i \tag{5.100}$$

$$e = K \cdot \dot{U} \tag{5.101}$$

From the figure above we can write the following equations based on Newton's law combined with Kirchhoff's law:

$$J\ddot{U} + b\dot{U} = \mathrm{T} \tag{5.102}$$

$$L\frac{di}{dt} + Ri = \mathrm{V} - \mathrm{K}\dot{\mathrm{U}} \tag{5.103}$$

Equation (5.102) describes the torque that can be generated by the DC motor. On the other hand, this torque has to balance the inverted pendulum and compensate the moment described by the equation (5.99). Therefore, combining these two equations we can obtain:

$$J\ddot{U} + b\dot{U} = \mathrm{m}_2 \mathrm{g} l_2 \vartheta_2 - m_2 l_2^2 \ddot{\vartheta}_2 \tag{5.104}$$

Equation (5.104) relates U - the desired position of the motor axis of the hip joint with ϑ_2 - the actual measured position of the trunk of the humanoid robot. Taking the Laplace

transformation of the equation (5.104), and simplifying, we can obtain the following transfer function:

$$\frac{\vartheta_2(s)}{U(s)} = \frac{-\frac{J}{m_2 l_2^2}s^2 - \frac{b}{ml^2}s}{s^2 - \frac{g}{l_2}} \tag{5.105}$$

In order to design the LQR controller let us transform (5.105) into a space state representation:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ \frac{g}{l_2} & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u \tag{5.106}$$

The output of the open loop system then is:

$$y = \begin{bmatrix} -\frac{Jg}{m_2 l_2^3} & -\frac{b}{m_2 l_2^2} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} - \frac{J}{m_2 l_2^2} u \tag{5.107}$$

The state space representation (5.106), (5.107) is a controllable canonical form that is important for the LQR controller design with the optimal control vector $u(t) = -\mathbf{K}\mathbf{x}(t)$. It is desired to keep the actual measured attitude angle ϑ_2 close to zero in order to maintain the trunk of the humanoid robot always in a vertical position as was discussed in the previous chapter. The block diagram showing the optimal attitude controller configuration for the inverted pendulum system is presented in the figure 5.37.

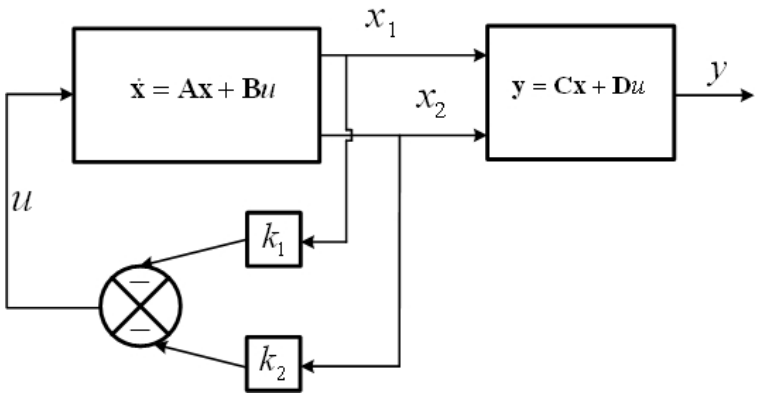


Fig. 5.37: Attitude LQR controller

The control system presented in figure 5.37 is a regulator system. In this case the controller maintains desired angle ϑ_2 of the pendulum close to zero. A command tracking so that the real humanoid robot hip joints could be positioned anywhere inside the motion pattern can be

achieved by adding an offset to the desired angle of the hip joint. It allows the LQR regulate the inverted pendulum about zero point. Figure 5.38 shows the performance of designed controller. Output $y = \vartheta_2$ versus t and control signal $u = U$ versus t when the system was placed in an arbitrary initial position of the trunk. It can be observed that approximately in 0.5 it returns to the desired position (zero).

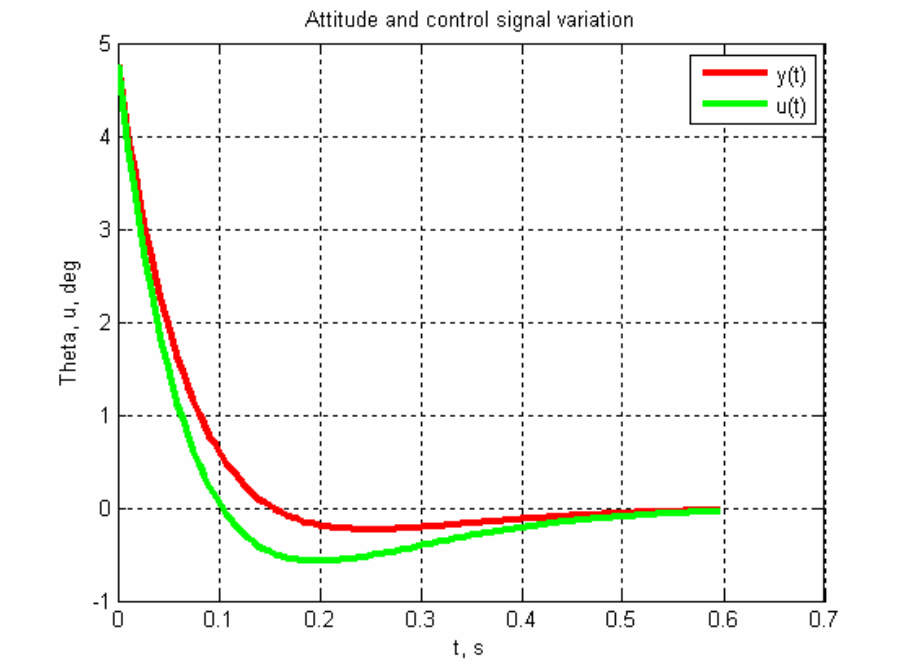


Fig. 5.38: Attitude and control signal variation

To conclude, it should be mentioned that the real 3D trunk's motion can be interpreted as a combination of two planar inverted pendulums in frontal and sigittal planes which have practically the same dynamics. Therefore the design of the Attitude controller presented in this section is useful for the frontal YZ plane case too.

5.6.4 Sensorial data acquisition and processing

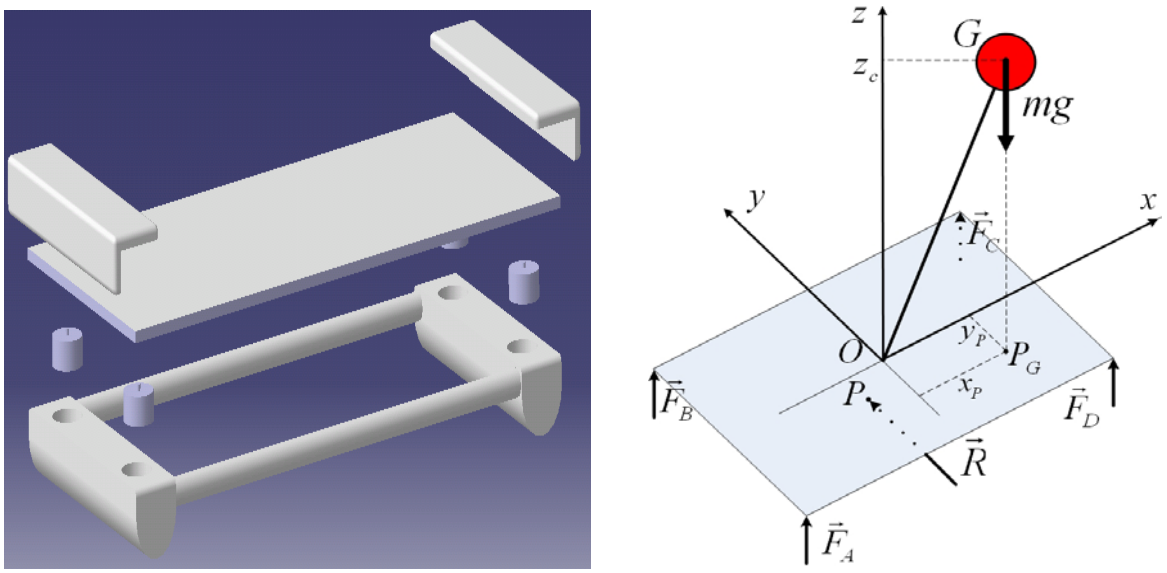
Previous sections described the design of LQR controllers to maintain the walking stability of the humanoid robot. From the other hand, it is necessary to provide the sampling from the real world to generate data that can be manipulated by a controller. Data acquisition typically involves acquisition of signals and waveforms from the real system and processing the signals to obtain desired information which will be used in the control process. The components of data acquisition systems include appropriate sensors that convert any measurement parameter to an electrical signal, which is acquired by data acquisition hardware. These signals should be processed in order to convert it into the measured value of the real system allowing its use in the control system. In our case, we need to measure and estimate two principal parameters characterizing the humanoid robot's walking. First of all it is the real Zero Moment Point used

for the ZMP control and then, the real inclination of humanoid's trunk from vertical position used for the Attitude control.

5.6.4.1 ZMP measurements.

A variety of sensors including load cells with strain gauges or six-axis force-torque sensors can be employed under the humanoid robot foot to obtain ZMP information. An ideal sensor should be light, add minimal thickness under the foot and should be reliable. Also, the issue of the cost effective solution is important. Therefore in this work both of these methods will be presented and compared.

The easy and cheap solution for the estimation of the real ZMP can be achieved by using 4 load cells located in each foot as is shown in figure 5.39(a) [Arbulu 2005].



a) b)

Fig. 5.39: ZMP measurement system. a) Mechanical configuration b) Forces distribution diagram

Figure 5.39(b) demonstrates the distribution of all forces affecting the foot platform. When the humanoid robot walks, load cells capture its total weight distributed over the four corners of the foot platform as $\mathbf{F_A}$, $\mathbf{F_B}$, $\mathbf{F_C}$ and $\mathbf{F_D}$. Considering that vertical components of inertial force acting on the COG are close to zero, total weight $\mathbf{W} = m\mathbf{g}$ is applied to the centre of gravity $\mathbf{G}$. When one foot supports the entire robot's body, the static force equilibrium is:

$$\mathbf{W} = \mathbf{F_A} + \mathbf{F_B} + \mathbf{F_C} + \mathbf{F_D} \tag{5.108}$$

Point P_G, there the resulting force is applied, is a precision projection of the center of gravity onto the support foot surface. Note that the point P where the resulting ground reaction force is applied is not the same point as P_G, although sometimes (in a static case when the humanoid robot is stopped) these two points coincide. Therefore, as this projection is not the same point as the real ZMP, in the previous section it was called a pseudo ZMP. If we do not take into account

inertial forces acting on the robot, pseudo ZMP can be considered as the real ZMP. As the humanoid robot walks with rather low speed (1 km/h), the point P_G can be taken as rough estimation of the real ZMP.

From the static equilibrium of moments $\sum M_Y = 0$ and $\sum M_X = 0$ we can get:

$$x_P = \frac{\mathbf{F_B} + \mathbf{F_C}}{\mathbf{W}} a \tag{5.109}$$

$$y_P = \frac{\mathbf{F_D} + \mathbf{F_C}}{\mathbf{W}} b \tag{5.110}$$

where x_P and y_p are coordinates of the COG projection in the coordinate system associated with the center of the supporting foot, a and b are known distance from the center of the foot to points where load sensors are located. Substituting total weight $\mathbf{W}$ by its meaning from equation (5.108) we get the way to compute the projection of the COG on the ground:

$$x_P = \frac{\mathbf{F_B} + \mathbf{F_C}}{\mathbf{F_A} + \mathbf{F_B} + \mathbf{F_C} + \mathbf{F_D}} a \tag{5.111}$$

$$y_P = \frac{\mathbf{F_D} + \mathbf{F_C}}{\mathbf{F_A} + \mathbf{F_B} + \mathbf{F_C} + \mathbf{F_D}} b \tag{5.112}$$

For the double support phase we can use the same proceedings with small variations. In this case the model will have two pendulums (legs) connected to the same point mass. Vertical force affecting the system is the same gravity although there will be eight points where the reaction is measured:

$$\mathbf{W} = \mathbf{F_A} + \mathbf{F_B} + \mathbf{F_C} + \mathbf{F_D} + \mathbf{F'_A} + \mathbf{F'_B} + \mathbf{F'_C} + \mathbf{F'_D} \tag{5.113}$$

In order to formulate the equilibrium of moments it is necessary to choose the reference system, different from coordinate systems associated with each foot. Usually this reference system is associated with the waist of the humanoid robot.

Formulating the equilibrium of moments $\sum M_Y = 0$ and $\sum M_X = 0$ it is possible to obtain the proceeding to compute the projection of the COG in the double support phase.

Let us recall equations (5.29), (5.30) describing the ZMP motion in the 3D-LIPM from the previous section 5.3.4:

$$x_{ZMP} = x_P - \frac{z_c}{g} \ddot{x}_P \tag{5.114}$$

$$y_{ZMP} = y_P - \frac{z_c}{g}\ddot{y}_P \tag{5.115}$$

here x_P and y_P are projections of the COG which can be computed as was shown above, $\ddot{x}_P$ and $\ddot{y}_P$ are the lateral COG accelerations and z_c is the plane where the motion of the COG is considered. Therefore, in order to compute the real ZMP it is necessary to add the measurement of the COG acceleration which can be provided by an accelerometer located in the COG of the robot (in practice it can be located in the trunk of the robot). Another unknown parameter is z_c - the height of COG with respect to the ground. Humanoid robots usually consist of well known parts such as arms and legs, feet and hands, a trunk, a neck and a head with given masses and sizes (Figure 5.40).

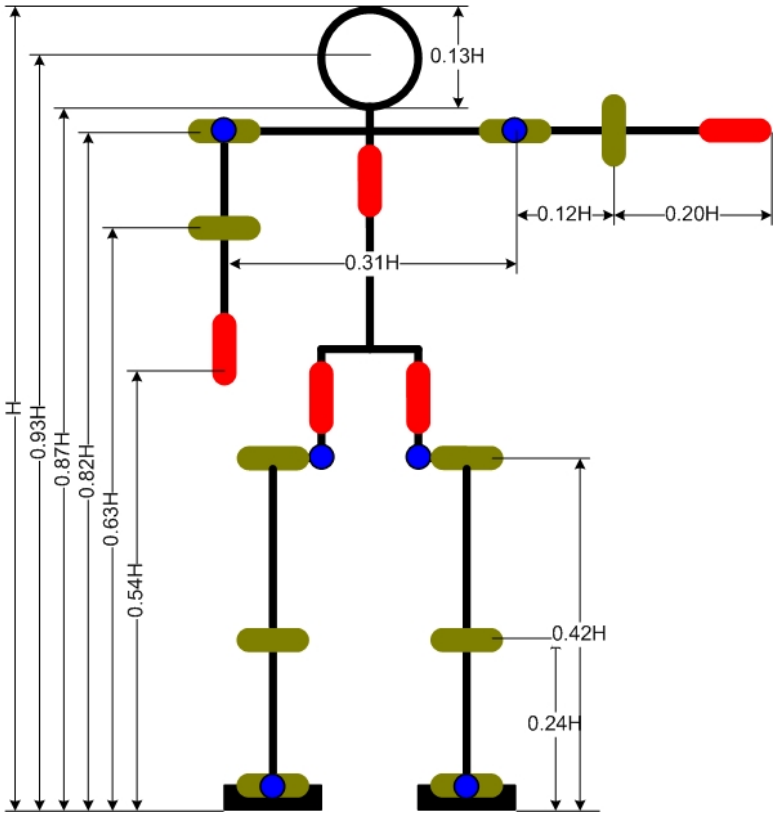


Fig. 5.40: Humanoid robot parts' dimension and distribution

Therefore, it is possible to compute the location of their COGs. Taking into account the symmetry of the robot's body (in the static case as well as during walking) the COG should always remain in the center of this symmetry. Finally, a total vertical COG position can be estimated from the following equation:

$$z_{COG} = z_c = \frac{1}{M}.\sum m_i.z_i = const \tag{5.116}$$

where M is a total mass of the robot, m_i is the mass and z_i is the vertical COG position of the i-th element of the humanoid's mechanical structure. Thus, combining load cells data with additional sensorial information (accelerometers) and the computing of the COG for the whole robot's body it is possible to obtain from equations (5.114) and (5.115) the real ZMP position.

Another way to get the real ZMP position is to use the 6-axis force-torque sensor which measure forces F_x, F_y, F_z and torques τ_x, τ_y, τ_z acting from the ground to the robot's foot. In addition, the foot should have a passive compliance to protect the sensor and the mechanical structure from the foot landing impact [Kajita 01]. Figure 5.41 illustrates the foot with mounted force-torque sensor.

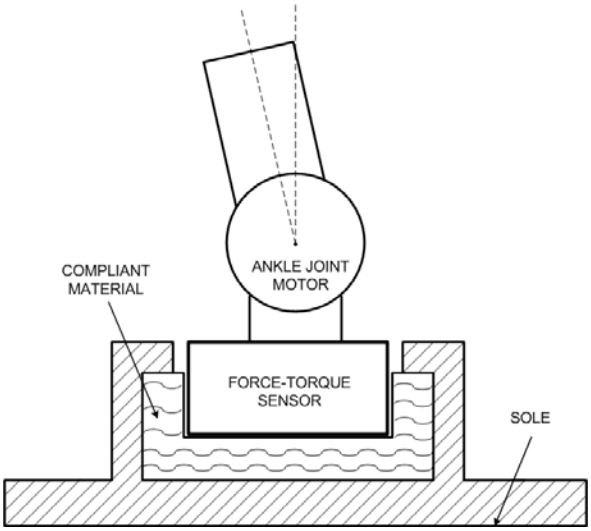


Fig. 5.41: Foot with mounted 6-axis force-torque sensor

From equations (5.27) and (5.28) presented in the previous section 5.3.3, and assuming that the COG always remains within the horizontal plain, the ZMP can be computed directly if the horizontal reaction moments in the contact point of the foot with a ground are known. In the real case, these moments are measured in some distance d from the real contact point (as is shown in the figure 5.42) because the force-torque sensor is separated from the ground by the compliance material and the sole. Therefore, for the simple support phase, when the entire robot's body is supported by, for example, the right leg (Figure 42 (a)) ZMP equations are transformed into:

$$xzmp_R = -\frac{\tau y_R - Fx_R \cdot d}{Fz_R} \tag{5.117}$$

$$yzmp_R = \frac{\tau x_R - Fy_R \cdot d}{Fz_R} \tag{5.118}$$

where $xzmp_R$ and $yzmp_R$ are coordinates of the real ZMP in the reference system related to a foot of the robot, $\tau x_R, \tau x_R, Fx_R, Fy_R, Fz_R$ are the data provided by force-torque sensors and d is

a vertical distance from the contact point on the sole of the foot and the sensor. In order to unify ZMP computing and interpretation it is necessary to transform equations (5.117) and (5.118) into the universal global reference coordinate system. Usually, this point is taken in the waist of the humanoid robot because of its relative similarity with the COG. Thus, the real ZMP of the robot in waist coordinates can be obtained by applying the inverse transformation:

$$X'zmp_R = T^{-1} \cdot Xzmp_R \tag{5.119}$$

where T is the total transformation matrix from the global reference frame (waist coordinate system) to a local reference frame (foot coordinate system) computed by the consecutive coordinates transformation in all kinematics chain.

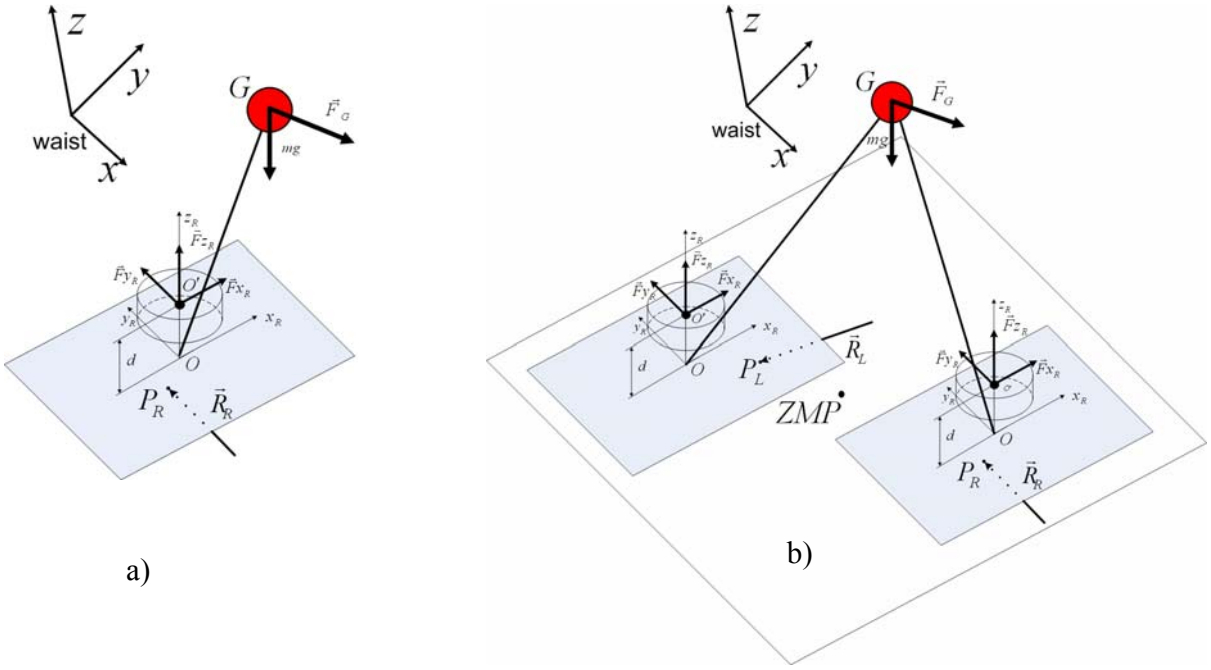


Fig. 5.42: Force-torque sensor implementation scheme a) Single support phase b) Double support phase

Thus, for the double support phase (Figure 5.42(b)) it is possible to compute the real ZMP in waist coordinates as a combination of forces and torques from both feet:

$$Xzmp = \frac{X'zmp_R \cdot Fz_R + X'zmp_L \cdot Fz_L}{Fz_R + Fz_L} \tag{5.120}$$

$$Yzmp = \frac{Y'zmp_R \cdot Fz_R + Y'zmp_L \cdot Fz_L}{Fz_R + Fz_L} \tag{5.121}$$

In conclusion, it should be mentioned that the use of load cells for estimating the ZMP is a cheap but not especially a reliable method because it requires a complement accelerometer

sensor. The combination of two electrical signals from different physical devices increases the estimation error and can be used only for experimental measurements. The 6-axis force-torque sensor is costly but reliable and easy to implement solution for estimating the ZMP position. As long as the development of humanoids remains very restricted area because of high costs and technological requirements, this sensor still seems to be the most appropriate solution.

5.6.4.2 ZMP data processing.

After the ZMP is computed from the sensorial data it should be processed in order to be used in the control system. Experiments show that the measured ZMP data is not very good because of high frequency vibrations related to compliance material or the mechanical structure flexion. In figure 5.45 the green curve shows the typical ZMP simulation computed from the force-torque sensor output. It can be clearly seen that the useful signal is affected by a high frequency noise. Therefore, the low pass filter should be implemented (Figure 5.43).

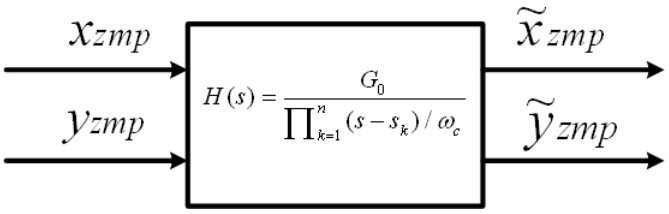


Fig. 5.43: Butterworth low-pass filter

The Butterworth filter was used because its frequency response is maximally flat (has no ripples) in the passband, and rolls off towards zero in the stopband. Also it is the only filter that maintains the same shape for higher orders (but with a steeper decline in the stopband) whereas other varieties of filters (Bessel, Chebyshev, elliptic) have different shapes at higher orders. Compared with a Chebyshev Type I/Type II filter or an elliptic filter, the Butterworth filter has a slower roll-off, and thus will require a higher order to implement a particular stopband specification. However, the second-order Butterworth filter was chosen and simulations proved its good applicability for ZMP data processing.

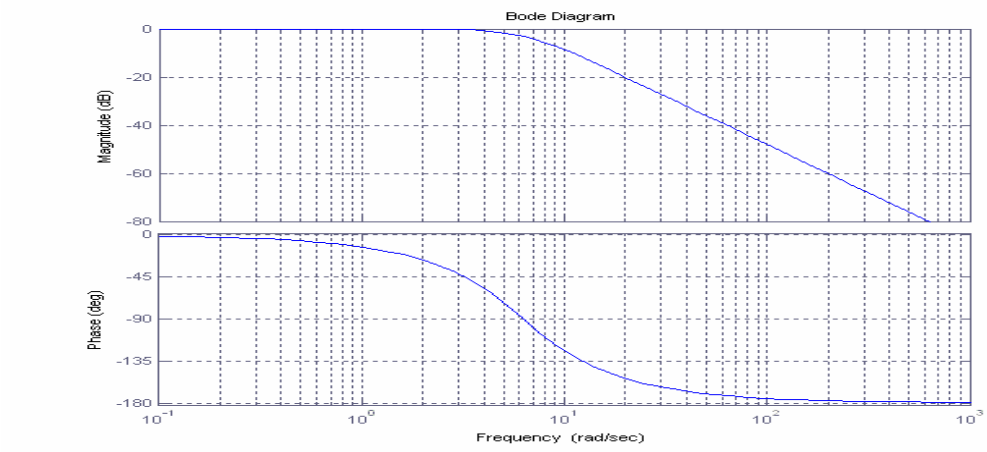


Fig. 5.44: Low-pass filter Bode diagram

The cutoff frequency of the filter was chosen as 1 Hz – the approximate frequency of the bipedal locomotion. As can be seen in figure 5.44, the response of the second order filter decreases at −12 dB per octave. The blue line in the figure 5.45 shows the performance of the designed second order Butterworth low-pass filter.

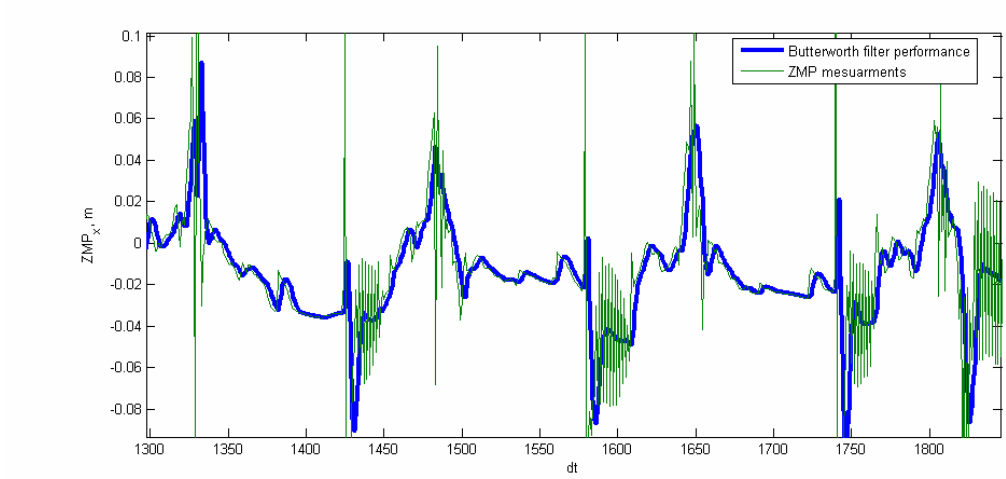


Fig. 5.45: Butterworth filter performance

Finally, measured, computed, and filtered ZMP is the input signal for the ZMP controller discussed in the previous section.

5.6.4.3 Attitude measurements.

As was described in the previous section, it is necessary to estimate and control the orientation of the humanoid's trunk to stabilize its bipedal locomotion. Figure 5.46 shows the angular description of the upper body of the robot.

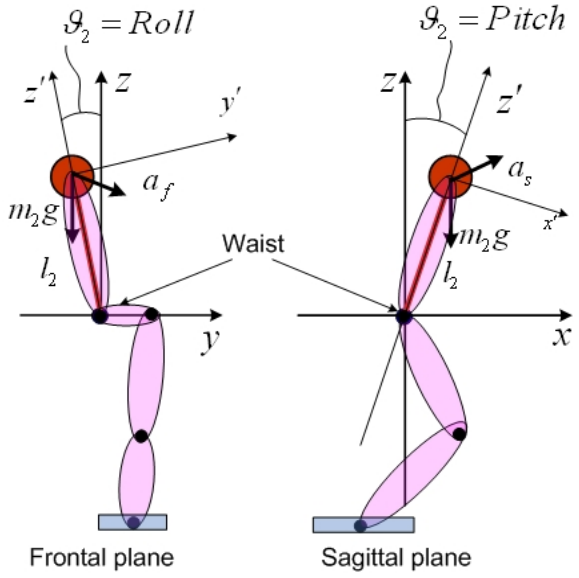


Fig. 5.46: Inertial reference system of the Upper body

In this figure two different planes of the robot's motion are presented. XZ and YZ are reference systems related to the waist of the robot (usually used as a main reference system to produce and control bipedal locomotion). $X'Z'$ and $Y'Z'$ are local reference systems associated with the trunk (usually these systems are associated with attitude sensors mounted in the trunk). The dynamical system is affected by gravitational g and translational a_s and a_f accelerations. The angle ϑ_2 characterizing the inclination of the upper body in the inertial system can be described as *Roll* in the frontal plane and *Pitch* in the sagittal plane. Therefore, the orientation of the upper body can be determined by the combination of inertial sensors such as an accelerometer and a gyroscope.

Accelerometers measure the linear acceleration of the object in the local inertial reference frame $X'Z'$ (since the accelerometers are fixed to the object and rotate with the object). Figure 5.47 presents the inertial reference plane of the inverted pendulum in the sagittal plane.

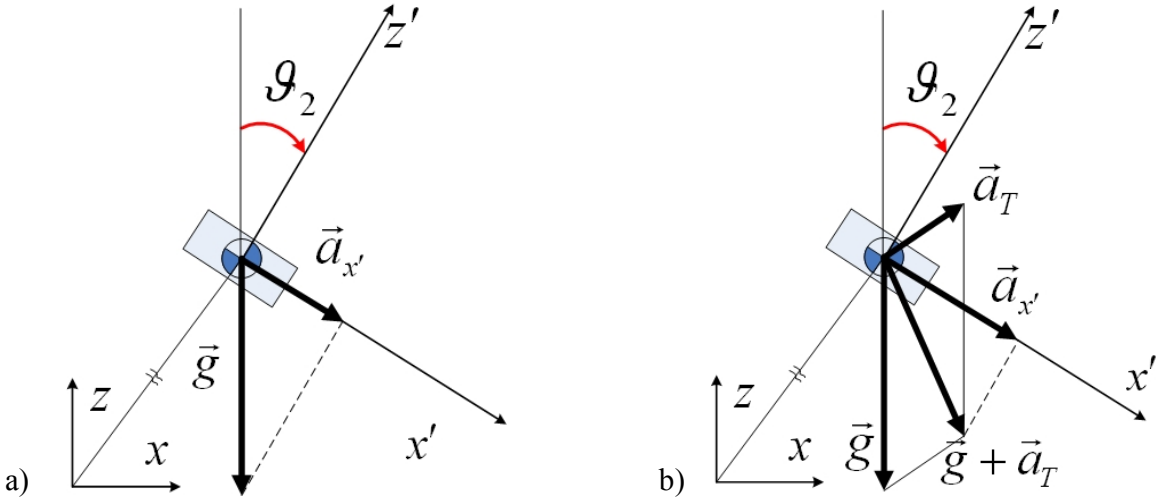


Fig. 5.47: Accelerometer measurements in sagittal plane a) static case b) dynamic case

In the static case, when translational accelerations does not affect the system (Figure 5.47(a)), the accelerometer measures only the projection of the gravity acceleration $\vec{g}$ on the axis X' of the local reference frame $X'Z'$. In this case, the attitude of the upper body can be computed by the following equations:

$$Pitch = \arcsin\left(\frac{a_{x'}}{g}\right) \tag{5.122}$$

$$Roll = \arcsin\left(\frac{a_{y'}}{g}\right) \tag{5.123}$$

where $a_{x'}$ and $a_{y'}$ are accelerations measured by the accelerometer in frontal and sagittal planes. In the real dynamical case, the motion of the humanoid robot generates the appearance of

translational accelerations $\vec{a}_T$ (Figure 5.47(b)). Thus, the accelerometer measuring $a_{x'}$ and $a_{y'}$, in fact, measures the projection of vector $\vec{g}+\vec{a}_T$ into the axis X' of the local reference frame $X'Z'$. Using equations (5.122) and (5.123) to compute inertial *Roll* and *Pitch* angles we will have an error in the attitude estimation. This error is more perceptible in high frequencies when the translational accelerations are higher. Therefore, a rate gyro as an additional source of angular data to compensate this effect is needed.

The rate gyro measures the rate of rotation, or in other words the angular velocity of the system in the inertial reference frame. The angle of orientation can be obtained through the integration of a sensor signal. Usually, rate gyros suffer a lot from temperature dependency bias. As we have to integrate the signal, small errors in the measurement of angular velocity are integrated into progressively larger errors in position that is known as a drift phenomenon. Figure 5.48 shows a typical drift in the integration of the gyro signal.

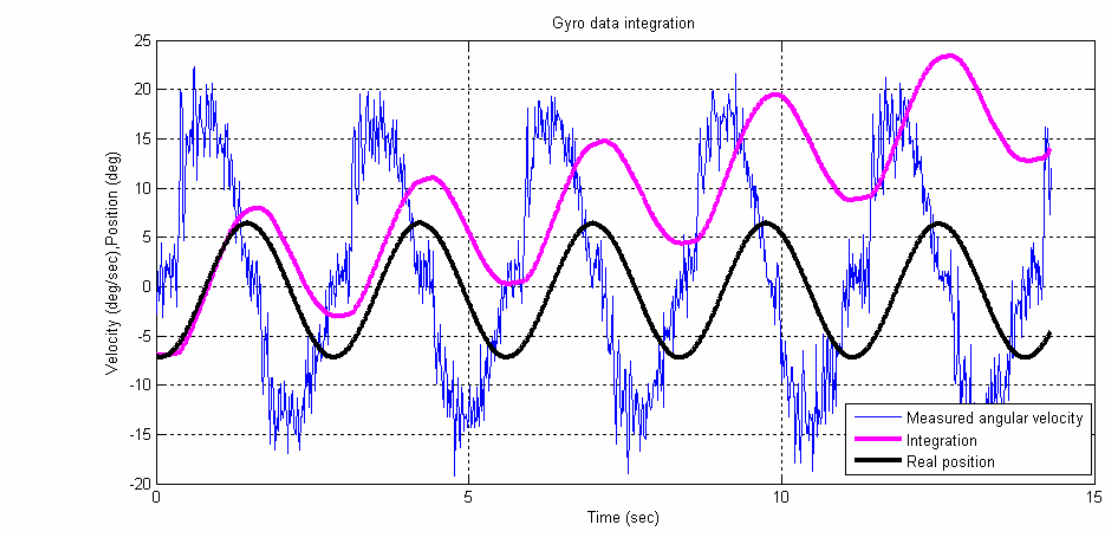


Fig. 5.48: An example of gyro's drift phenomenon

Finally, it can be concluded that measured inertial data obtained from both sensors should be combined. By this, it is possible to compensate the error in the estimation of inertial angles due to translational accelerations by gyro data and the error in the integration of angular velocity because of the drift phenomenon by accelerometer data. In following section two compensational methods will be proposed.

5.6.4.4 Attitude data processing.

In order to eliminate measurement noise and computing errors discussed in the previous section the data provided by inertial sensors should be processed by a filtering system. In this work two different systems such as the complementary High-Low Pass filter and the Kalman filter will be presented and compared and finally a combining method will be proposed.

The basic idea of the complementary High-Low Pass filter is to combine the outputs of the accelerometer and a rate gyro to obtain an acceptable estimation of the orientation of the

humanoid's trunk. The estimation of the orientation $\hat{\vartheta}_2$ is obtained as the sum of the sensorial data split into two branches (Figure 5.49). One of them is the orientation computed by equations (5.122) and (5.123) from accelerometer measurements and the second one is the integration of measurements of the rate gyro. The accelerometer feeds its output signal $\vartheta_{2m}(s)$ directly into the filter $G_A(s)$ which contributes to the estimation $\hat{\vartheta}_2$ in a low frequency domain, thus, eliminating the error related with the influence of translational accelerations on the accelerometer measurements. The gyro provides measurements of rotational velocity of the system $\dot{\vartheta}_{2m}(s)$ which is integrated and yielded into the filter $G_G(s)$. This filter contributes to the estimation of $\hat{\vartheta}_2$ in high frequency domain, thus, eliminating the error because of the drift of the gyro.

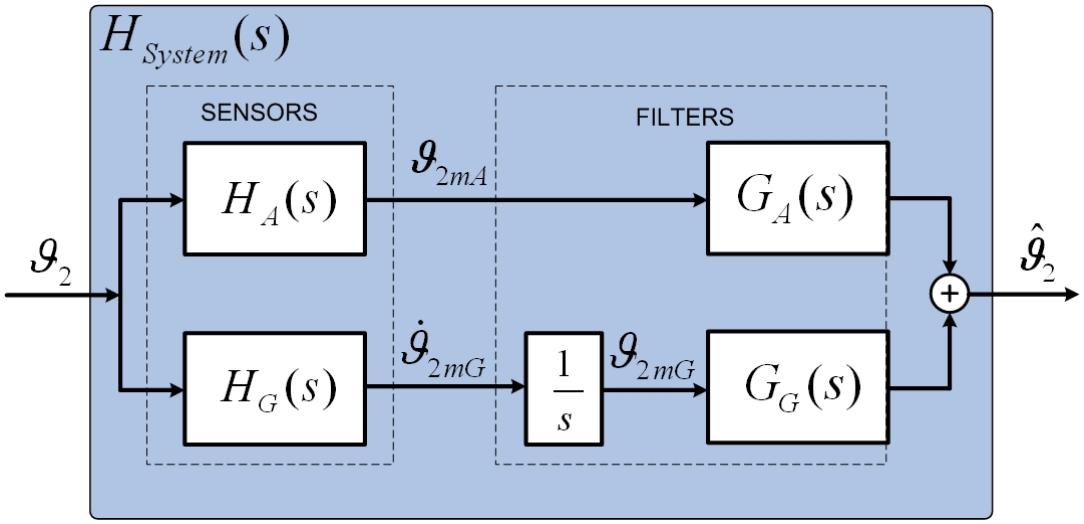


Fig. 5.49: Attitude estimation system

The estimated angle $\hat{\vartheta}_2$ should manifest the same dynamics as the real angle of orientation ϑ_2. In that case, the transfer function $H_{System}(s)$ of the whole attitude estimation system should fulfill following condition:

$$H_{System}(s) = \frac{\vartheta_2}{\hat{\vartheta}_2} = 1 \tag{5.124}$$

As the transfer function $H_{System}(s)$ is the superposition of all components of the estimation system, we can state:

$$H_A(s)G_A(s) + H_G(s)\frac{1}{s}G_G(s) = 1 \tag{5.125}$$

If we assume ideal sensors (filter functions will be designed in order to compensate the non ideality of the real sensor) we can take:

$$H_A(s) = 1 \tag{5.126}$$

$$H_G(s) = s \tag{5.127}$$

Thus, the equation (5.125) becomes:

$$G_A(s) + G_G(s) = 1 \tag{5.128}$$

The condition posed by the equation (5.128) leads to the infinite number of possible filter functions $G_A(s)$ and $G_G(s)$. Therefore, the basic requirements for adequate filters can be formulated [Baerveldt 97]: The overall system should exhibit constant amplification and small phase loss up to frequencies well above the cut-off frequency of the accelerometer. In order to keep the sensitivity to offsets of the gyro to a minimum, the accelerometer should be used in the widest possible ranges of frequencies. Finally, the number of design parameters should be small in order to simplify the tuning and implementation of filters. In order to keep the number of filter design parameters small, the second order filter with a double pole can be chosen:

$$G_A(s) = \frac{2\tau\, s + 1}{(\tau\, s + 1)^2} \tag{129}$$

$$G_G(s) = \frac{\tau^2 s^2}{(\tau\, s + 1)^2} \tag{130}$$

where τ is a filter time constant. The filter $G_A(s)$ filtering the signal provided by the accelerometer is a first order low-pass filter (the response decreases -20 dB per decade) in series with a lead filter. The filter $G_G(s)$ filtering the integrated gyro signal is a second order high-pass filter (-40 dB per decade). The filter time constant τ is the same for the low-pass and the high pass filters, thus, it is the only design parameter. A high time constant has the advantage of the emphasizing of a good dynamics of the gyro. At the same time, it leads to weighting the accelerations high, thus making the estimation sensitive against linear accelerations [Loffler 2004]. In addition, the influence of the time-varying bias and drift of the gyro increases. Therefore, a time constant should be chosen as low as possible. However, as we assumed to have ideal sensors the time constant should be chosen above the time constants of the accelerometer and the gyro. Figure 5.50 shows the Bode plot of the accelerometer $G_A(s)$ and gyro $G_G(s)$ filters with the filter constant time $\tau = 1$ s.

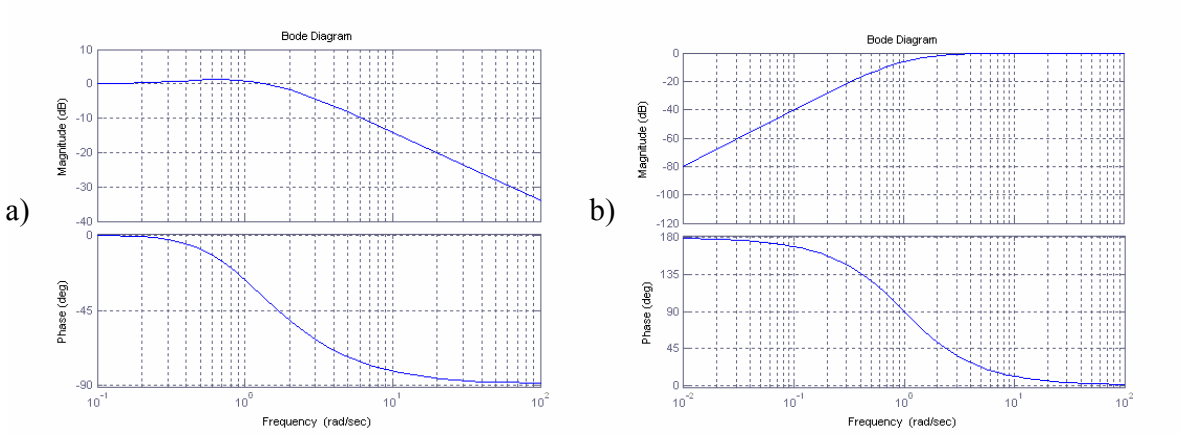


Fig. 5.50: Bode plot a) Accelerometer filter b) Gyro filter

Figure 5.51 shows the simulation plot of the estimated orientation $\hat{\vartheta}_2$ with the complementary filter.

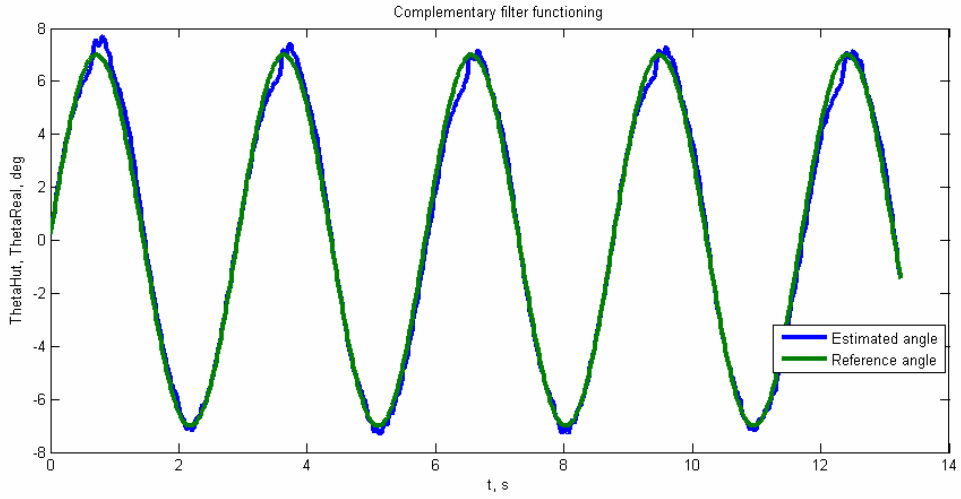


Fig. 5.51: Complementary filter performance

The excitation reference input of the system is a sinusoidal signal of 0.35 Hz. As can be observed, the complementary filter shows a good estimation of the orientation on low frequencies of the input signal. The almost neglible differences which are visible at the peaks of the signal are probably due to calibration errors of the sensors and can be removed by better calibration. However, further experiment showed that the functioning of the designed complimentary filter deteriorates with the rising of the frequency of the input signal. This occurs because the low-pass branch is more and more corrupted by translational accelerations. As walking frequency of a humanoid can exceed 1 Hz, it is necessary to add another method to estimate the exact orientation of the trunk of the humanoid robot.

One method which can be used in addition to the complementary High-Low pass filter is the Kalman filter. The Kalman filter is an efficient recursive filter that estimates the state of a dynamic system from a series of incomplete and noisy measurements. The Kalman filter uses the dynamics of the dynamical system, which governs its evolution over time, in order to remove the effects of noise, measurement and computational errors. It allows for a good estimation of the system's parameters at the present time (filtering), at a future time (prediction), or at a time in the past (interpolation or smoothing). In order to use this filter to estimate the internal state of a process having only a sequence of noisy measurements, it is necessary to model the process in accordance with the framework of the Kalman filter. Then, it addresses the general problem of trying to estimate the state x_k of a discrete-time controlled process that is governed by the linear stochastic difference equation [Welch 2004]:

$$x_k = Ax_{k-1} + Bu_{k-1} + w_{k-1} \tag{5.131}$$

The $n \times n$ matrix A in the difference equation (5.131) relates the state at the previous time step $k-1$ to the state at the current step k, in the absence of either a driving function or process noise. The $n \times l$ matrix B relates the optional control input u to the state x. In practice A and B might change with each time step, but here we assume it is constant. w_k is the process noise which is assumed to be drawn from a zero mean multivariate normal distribution with covariance Q:

$$p(w_k) \sim N(0, Q) \tag{5.132}$$

At time k an observation (or measurement) z_k of the true state x_k is made according to:

$$z_k = H_k x_k + v_k \tag{5.133}$$

where the $m \times n$ matrix H is the observation model which maps the true state space into the observed space (relates state x_k to the measurement z_k). In practice H might change with each time step or measurement, but here we assume it is constant. v_k is the observation noise which is assumed to be zero mean Gaussian white noise with covariance R:

$$p(v_k) \sim N(0, R) \tag{5.134}$$

In practice, the process noise covariance Q and measurement noise covariance R matrices might change with each time step or measurement, however here we assume they are constant [Welch 2004]. Also, the initial state, and the noise vectors at each step $\{x_0, w_1, \ldots, w_k, v_1, \ldots, v_k\}$ are all assumed to be mutually independent.

Now, considering the motion of the upper body as the motion of the inverted pendulum (Fig. 5.46) as was described above, we can describe linear equations of the process to be estimated. Writing the equations of rotational kinematics of the inverted pendulum we have:

$$\vartheta_2(t) = \vartheta_2(0) + \dot{\vartheta}_2(0)t + \frac{1}{2}\alpha t \tag{5.135}$$

$$\dot{\vartheta}_2(t) = \dot{\vartheta}_2(0) + \alpha t \tag{5.136}$$

where $\vartheta_2(t)$ is the actual angular position and $\dot{\vartheta}_2(t)$ the actual angular velocity of the inverted pendulum, and $\vartheta_2(0)$, $\dot{\vartheta}_2(0)$ its initial values, α is an angular acceleration and t is a time vector. The orientation ϑ_2 and angular velocity $\dot{\vartheta}_2$ of the trunk can be represented in the linear state space as:

$$x_k = \begin{bmatrix} x_{1k} \\ x_{2k} \end{bmatrix} = \begin{bmatrix} \vartheta_2 \\ \dot{\vartheta}_2 \end{bmatrix} \tag{5.137}$$

Assuming that between the $(k-1)^{th}$ and k^{th} time steps the pendulum (trunk of a humanoid robot) undergoes a random time varying angular acceleration α_k that is normally distributed, with mean 0 and standard deviation σ_a, from (5.131) and (5.135)-(5.137) we conclude that:

$$x_k = Ax_{k-1} + G\alpha_k \tag{5.138}$$

In the previous equation we ignore B and u_{k-1} supposing that there is no control on the trunk. The state transition matrix A is then:

$$A = \begin{bmatrix} 1 & \Delta T \\ 0 & 1 \end{bmatrix} \tag{5.139}$$

And the random process noise matrix G is:

$$G = \begin{bmatrix} \frac{\Delta T^2}{2} \\ \Delta T \end{bmatrix} \tag{5.140}$$

where ΔT is a time step between $(k-1)^{th}$ and k^{th} states of the system. In practice, this time is taken as sample time of the data acquisition system (will be established in the chapter 6).

We can find that since a standard deviation σ_a is a scalar value, a process noise covariance can be found as:

$$Q = \text{cov}(G\alpha) = E\left[(G\alpha)(G\alpha)^T\right] = GE[\alpha^2]G^T = G[\alpha_a^2]G^T = \sigma_a^2 GG^T = \sigma_a^2 \begin{bmatrix} \frac{\Delta T^4}{4} & \frac{\Delta T^3}{2} \\ \frac{\Delta T^3}{2} & T^2 \end{bmatrix} \tag{5.141}$$

At each time step, a noisy measurement by the sensorial system (gyros and accelerometers) and computation of the true position of the trunk can be made as was shown bellow. In this work we suppose that the noise is normally distributed, with mean 0 and standard deviation σ_z. Therefore:

$$z_k = Hx_k + v_k \tag{5.142}$$

where H describes how the measured data relates (linearly) to the state:

$$H = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{5.143}$$

and measurements noise covariance R is:

$$R = E\left[v_k v_k^T\right] = \sigma_z^2 v_k v_k^T = \sigma_z^2 \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{5.144}$$

After we have the process and measurement models described, we need to formulate an estimation algorithm such that the following statistical conditions hold. The expected value of the estimate should be equal to the expected value of the state. That is, on average, the estimation of the state will equal the true state. We need an estimation algorithm that minimizes the expected value of the square of the estimation error. That is, on average, the algorithm gives the smallest possible estimation error [Simon 2006]. The Kalman filter is the estimation algorithm which satisfies these criteria. There are many alternative ways to formulate the Kalman filter equations. As the Kalman filter estimates a process by using a form of feedback control: the filter estimates the process state at some time and then obtains feedback in the form of (noisy) measurements therefore the equations for the Kalman filter conditionally can be divided into two groups: time update equations and measurement update equations [Maybeck 99] and [Welch 2004]. The time update equations are responsible for projecting forward (in time) the current state and error covariance estimates to obtain the *a priori* estimates for the next time step.

$$\hat{x}_x^- = A\hat{x}_{k-1} + Bu_k \tag{5.145}$$

$$P_k^- = AP_{k-1}A^T + Q \tag{5.146}$$

where $\hat{x}_x^-$ is (note the "super minus") is the *a priori* state estimate at step k given knowledge of the process prior to step k, and $\hat{x}_{k-1}$ is *a posteriori* state estimate at step $k-1$. Then, P_k^- is the *a priori* estimate error covariance at step k and P_{k-1} is the *a posteriori* estimate error covariance at previous $k-1$ step.

The measurement update equations are responsible for the feedback - i.e. for incorporating a new measurement into the *a priori* estimate to obtain an improved *a posteriori* estimate [Welch 2004]:

$$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1} \tag{5.147}$$

$$\hat{x}_k = \hat{x}_k^- + K_k (z_k - H\hat{x}_k^-) \tag{5.148}$$

$$P_k = (I - K_k H) P_k^- \tag{5.149}$$

The first task during the measurement update is to compute the Kalman gain K_k. The next step is to actually measure the process to obtain z_k, and then to generate an *a posteriori* state estimate by incorporating the measurement as in equation (5.148). The final step is to obtain an *a posteriori* error covariance estimate via equation (5.149). The difference $(z_k - H\hat{x}_k^-)$ in equation (5.148) is called the measurement innovation. The innovation reflects the discrepancy between the predicted measurement and the actual measurement. An innovation of zero means that the two are in complete agreement.

As we know that the initial starting state of the inverted pendulum (in this case we suppose that the trunk of the humanoid robot is in vertical position at the first moment) so we initialize:

$$\hat{x}_x^- = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{5.150}$$

and to tell the filter that we know the exact position, we give it a zero covariance matrix:

$$P_k^- = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{5.151}$$

After the initialization of the initial state, the recursive Kalman estimation mechanism can be started. The time update equations can be thought of as predictor equations, while the measurement update equations can be thought of as corrector equations.

After each time and measurement update pair, the process is repeated with the previous *a posteriori* estimates used to project or predict the new *a priori* estimates. This recursive nature is one of the very attractive features of the Kalman filter - it makes practical implementations much more feasible than an implementation of other filters. The Kalman filter instead recursively conditions the current estimate on all of the past measurements.

The practical implementation of the filter requires the knowledge of two basic design parameters. They are process noise and measuring noise standard deviations (σ_a and σ_z) which are used for computing measurement covariance and noise covariance. Usually it can be measured prior to operation of the filter. Measuring the measurement error covariance R is practically possible. It is necessary to make some off-line experimental measurements in order to determine the variance of the measurement noise. The determination of the process noise is generally more difficult task. As we usually do not have the ability to directly observe the process we should make estimations. Sometimes a relatively simple process model can produce acceptable results. Nevertheless, in this case it is difficult to be sure that the process

measurements are reliable. In either case, whether or not we have a rational basis for choosing the parameters, the desired filter performance can be obtained by previous off-line tuning of the filter parameters σ_a and σ_z. Moreover, the real dynamical system may not exactly fit the model which was described by equations (5.135) and (5.136); however, because the Kalman filter is designed to operate in the presence of noise, an approximate fit is often good enough.

Figure 5.52 shows the results of the Kalman filtering of the data measured by gyros and accelerometers mounted on the top of the inverted pendulum.

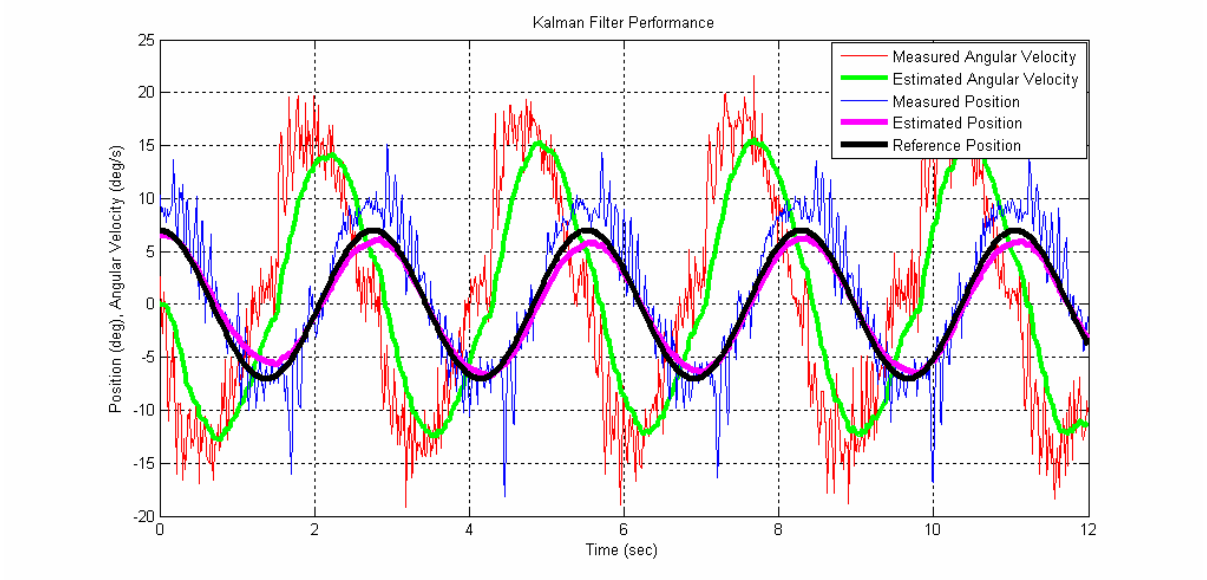


Fig. 5.52: Kalman Filter Performance

The excitation reference input of the system as in the previous case is a sinusoidal signal of 0.35 Hz. As can be observed, the Kalman filter shows rather good estimation of the orientation. Nevertheless, the quality of estimation depends on the initial tuning of the filter. The measurement error (in particular) does not remain constant. Also, the process noise is sometimes changed dynamically during filter operation in order to adjust it to different dynamics. Therefore the filter can be tuned appropriately only for a part of the operational frequency range. In this tuned range it reflects the dynamics properly and makes a reliable estimation of the process.

Finally, in order to complement the frequency range of the attitude filtering system, the Kalman filter can be tuned for the estimation on the high frequency range where the High-Low pass filter presented above has problems related with lateral accelerations of the accelerometer.

Thus, the attitude data processing system to be implemented in the stabilizer of humanoid robot takes the form presented in the figure 5.53.

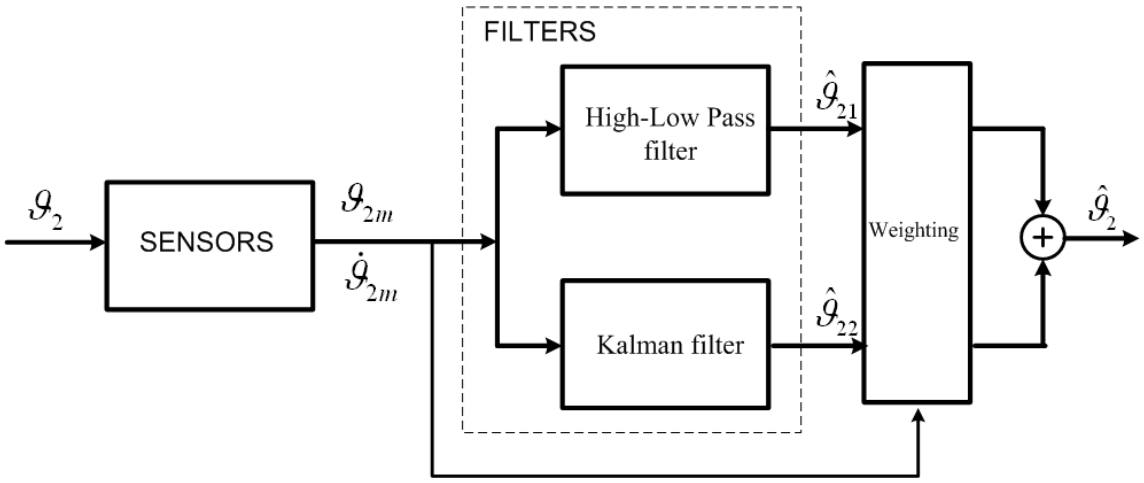


Fig. 5.53: Attitude estimation system

The real attitude of the trunk ϑ_2 is measured by the sensorial system (gyros and accelerometers) of the humanoid robot. Measured and computed values of the angular position ϑ_{2m} and angular velocity $\dot{\vartheta}_{2m}$ of the trunk are processed by two filters simultaneously. After this, the resulting estimation is obtained by weighting both estimations of the attitude. The High-Low pass complementary filter has more weight in a low frequency domain while the Kalman filter is tuned to have more reliable results in a high frequency domain. The final estimation $\hat{\vartheta}_2$ is the sum of both signals. The data processing should be made for both planes (frontal and sagittal) thus obtaining *Roll* and *Pitch* angles of the attitude. These angular estimations will be used in the attitude control of the humanoid robot.

5.6.5 Detailed Stabilizer architecture

The general stabilizer structure was presented in the figure 5.14. After all parts were developed in detail in previous sections let us present the detailed control architecture (Figure 5.54).

The sensorial system of the robot consisting of two six-axis force-torques should provide the controller with the real distribution of the forces and torques F_x, F_y, F_z, τ_x, τ_y, τ_z at the contact point of the foot with the ground. While the 3-axis Gyro and Accelerometer provide the measurements of the angular position ϑ_m and angular velocity $\dot{\vartheta}_m$ of the upper body (trunk) of the robot in the frontal and sagittal planes (Roll and Pitch). After the actual ZMP position x_{ZMP}, y_{ZMP} is computed by the ZMP Computational module and the real attitude is estimated in the Attitude Estimation module, the ZMP Δx_{ZMP}, Δy_{ZMP} and Attitude $\Delta\vartheta$ errors can be estimated. These errors are the input data for the Stabilizer. The Stabilizer is designed as a decoupled controller. It controls the error in ZMP and Attitude positioning of the humanoid robot by the motion of the ankle and hip joints as was discussed above.

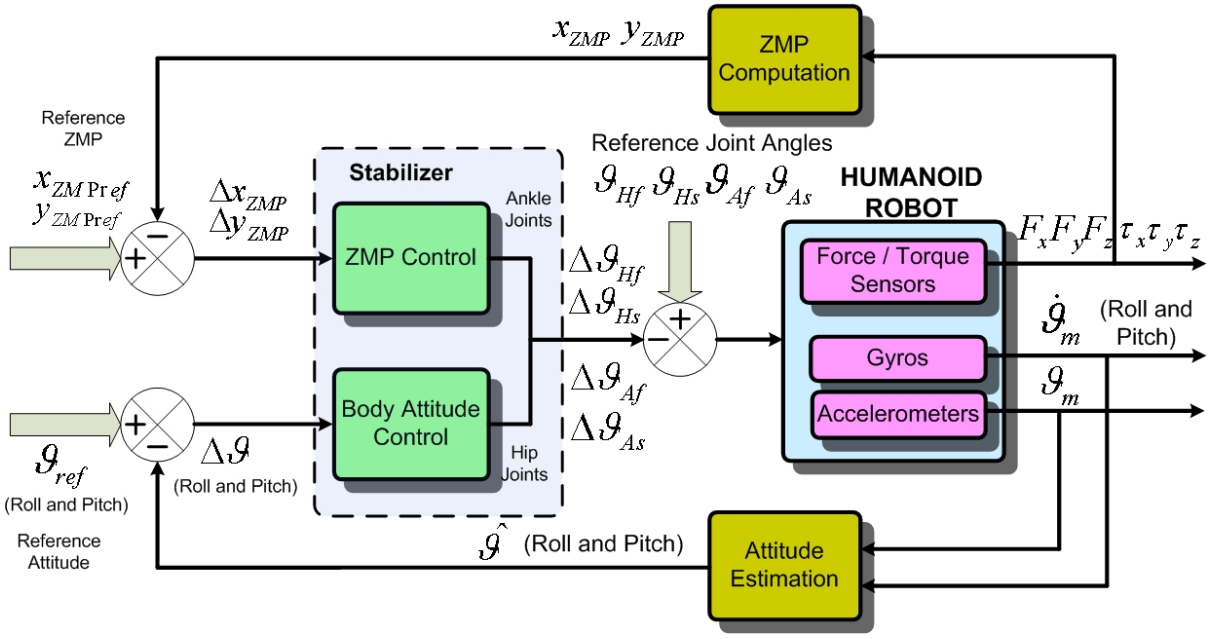


Fig. 5.54: Stabilizer architecture

Finally, the compensational motion of the ankle $\Delta\vartheta_{Af}$, $\Delta\vartheta_{As}$ and hip $\Delta\vartheta_{Hf}$, $\Delta\vartheta_{Hs}$ joints in the frontal and sagittal planes should be superposed with their reference trajectories ϑ_{Af}, ϑ_{As}, ϑ_{Hf}, ϑ_{Hs} which are responsible for the entire walking process. In this way, new motion patterns for ankle and hip joints are generated. The implementation of the decoupled stabilizer provides fast and easy control of the walking stability of the humanoid. All changes are applied to ankle and hip joints eliminating the need for inverse kinematics computation. Finally, the physical realization of the stabilizer is a software module integrated into the control system of the humanoid robot. This implementation as well as provided simulation experiments with the designed stabilization system for humanoid robots will be discussed in the chapter 7.

5.7 Conclusions

The most important criterion for determining the stable locomotion of a humanoid robot is the ZMP concept. This concept is currently widely used as a basic method for generating stable bipedal locomotion. One of the possible ways to generate stable motion patterns is the 3D-LIPM. This method uses the dynamics of the inverted pendulum in order to model bipedal walking.

Although planned motion patterns satisfy the stability constraints, some errors caused by the dynamics, irregularity of the terrain or some external forces can cause the humanoid robot to fall over. Moreover, the flexion of the mechanical structure of the humanoid robot also is the large source of errors affecting its walking. To reduce the influence of these errors on walking stability, the new method of online motion patterns modification for the stabilization control of the humanoid robot was proposed.

As the humanoid robot's dynamics under the 3D-LIPM is governed by the ZMP equations, the main control variables are ZMP and COG positions. In this way, the stabilizer should provide the control of both of these parameters. Further studies of the humanoid's locomotion showed that the most effective way to control ZMP is the correction of the trajectory of the supporting ankle joint. And the best way to control COG position is maintaining the attitude of the Upper body by means of the correction of hip's trajectories. This control can be modeled as a double inverted pendulum. The mathematical model of the stabilization control of a humanoid robot was developed and the controller designed. However, the coupled control of the double inverted pendulum in practice has some inconveniences. The state space representation of the system is a 4x2 equation. It means that there are 4 states and 2 control inputs. Moreover, the main control variable is the motor torque, but most of contemporary humanoid robots are driven by DC motors with position control where torque control cannot be implemented.

In order to simplify the stabilization control of the humanoid robot, the new decoupled approach was proposed. It considers that ZMP and Attitude parts can be implemented as totally independent and the influence of one on the other is small and can be compensated by the controller. In this way it is possible to divide the state space representation of the system into two 2x1 equations (2 states and 1 input). Moreover, in each of the control equations the torque control input can be replaced by the reference position that is a typical control input for conventional DC motors implemented in a majority of contemporary humanoid robots. Finally, the structure of the decoupled stabilizer consisted of independent ZMP and Attitude controls was proposed and LQR controllers were designed.

After control law is designed, it is necessary to provide the sampling of the real world to generate data that can be manipulated by a controller. Two principal parameters characterizing the humanoid robot's walking should be estimated. They are the real ZMP and the Attitude.

A variety of sensors can be used for ZMP measurements. In this work we proposed two principal methods for measuring and computing the real ZMP – 4 load cells and 6-axes force-torque sensors. After the ZMP is computed from the sensorial data, it should be processed in order to be used in the control system. The design of the low pass Butterworth filter for ZMP data was also proposed.

The Attitude of the humanoid's trunk can be measured by combining the data from two types of inertial sensors – gyroscopes and accelerometers. Both of these have problems with the accuracy of measurements or estimation. Thus, the new method, combining the data from two filtering systems (High-Low pass and Kalman filters) was developed.

Finally, the structure of the Stabilizer providing simple and effective way for controlling humanoid walking stability was proposed. Chapter 7 will present and discuss implementation and experiments carried out with designed stabilization system for humanoid robots.

Chapter 6

Open architecture for humanoid motion control

CHAPTER 6

Open architecture for humanoid motion control

6.1 Introduction

As mentioned in chapter 3, the physical architecture design primarily refers to the software and hardware framework that controls the robot. The development of software and hardware modules and the communication among them begins to define the physical level of control architecture.

Humanoid robotic systems are complex and tend to be difficult to develop. They integrate multiple sensors, have many degrees of freedom and must coordinate different hard real-time systems with systems which have no real-time deadlines. System developers have typically relied upon robotic architectures to guide the construction of robotic devices and to provide computational services (e.g., communications, processing, etc.) to subsystems and components [Dowling 95]. These architectures, however, are usually task and domain specific and are not sustainable for a broad range of applications. For example, an architecture well suited for direct teleoperation is not applicable for supervisory control or for autonomous use. In the same way, motion control architecture defined in chapter 3 as the bottom level of global control architecture of a humanoid robot is a specific case of architectures and therefore, can be clearly defined. In chapter 3, which analyzes the general requirements for motion control, autonomous distributed hierarchical control architecture was proposed for implementation at the physical level (software and hardware systems) of humanoid robot development.

Before starting with the detailed description of the hardware and software systems in this chapter, let us pose some requirements for the development process of the architecture. It is important to develop the architecture (software and hardware) in a way that satisfies the following requirements:

- The system should consist of modules in order to fulfill the requirements of a distributed system. Each of them deals with a particular task in a limited local environment (modularity).
- The developer can easily add, remove and update the modules (scalability).
- The architecture should allow the developer to progressively implement different modules.

- The system represents relations among the modules and robot tasks on the module network (distribution).
- The system should be able to adjust the execution order of the modules automatically according to the different working conditions the robot faces. This helps to compensate for any incomplete module implementation on the part of the developer.

Following these requirements it is possible to simplify the motion control architecture design process for a humanoid robot and finally, increase its effectiveness.

Another aspect of the design process is the compatibility of components and tools. A closed componential base doesn't allow for easy upgrades, improvements of control systems or the development of new modules and elements. Open architecture is a type of hardware or software architecture that allows adding, upgrading and swapping of components as the user wishes without any additional permission or/and payment of fees to the developer. The development of humanoids nowadays suffers from the absence of openness in systems due to the commercial status of most of the projects (Honda ASIMO, Toyota, etc.). Nevertheless, the author defends open principles for architecture design to accelerate progress in the field of humanoid robotics.

This chapter is organized as follows. First of all, some observations on openness of system architecture are presented. After this, a detailed hardware and software design of a motion control system for humanoid robot will be provided.

6.2 Open architecture

6.2.1 Definition

Before starting with the design of the motion control architecture let us clarify the concept of open architecture.

There are many different definitions of an open system. An open system according to the definition given by the Software Engineering Institute of the Carnegie Mellon University is a collection of interacting software, hardware, and human components designed to satisfy stated needs with interface specifications of its components that are [Oberndorf 97]:

- fully defined
- available to the public
- maintained according to group consensus
- able to insure that implementation of the components conforms to the interface specifications

As with the definition of an open system, there are various definitions of the concept of open system architecture. Open architecture is produced by an open systems approach and employs open systems adequate specifications and standards. In another words, open architecture is a system architecture in which the components, both hardware and software, are specified in an open manner and these specifications are public. This definition includes standards officially approved by international bodies, as well as private architectures whose specifications are made

public by the designers. The concept of closed or proprietary architectures is opposed to the open architecture paradigm.

More specifically, an open architecture is a system representation which:

- maps the functionality of the system onto both - hardware and software components,
- links the software architecture with the hardware architecture,
- establishes possible means of human interaction with these two components,
- establishes specifications of each of the system's components.

Open architecture is characterized by the following [Meyers 97]: It is well defined, widely used, and contains non-proprietary interfaces/protocols. The open system uses standards developed/adopted by industrially recognized standards bodies. Open architecture defines all aspects of the system interfaces to facilitate new or additional system capabilities for a wide range of applications with explicit provision for expansion or upgrading through the incorporation of additional or higher performance elements with minimal impact on the system.

6.2.2 Open software architecture

An open software architecture can be defined as a systems' ability to provide a software environment that provides not only an external hook to interface to remote devices and/or equipment and pass data, but also an inherent internal ability to empower the user to enhance, add, and replace any internal function, configuration tools, and editors [Kramer 93]. Really open software architecture literally puts the user on the same level with the software vendors and development groups.

An open software architecture, first of all, means the development and implementation of open source software. Open source software is software for which the human-readable source code is made available under a copyright license (or arrangement such as the public domain) that meets the Open Source Definition [Perens 99]. This permits users to use, change, and improve the software, and to redistribute it in modified or unmodified form. It is often developed in a public, collaborative manner. Open source software is the most prominent example of open source development and often compared to user generated content [Verts 2008].

Open source appeared during the 1970s when a "free" version of the widely used "Unix" operating system was developed by Richard Stallman. The resulting "GNU" (a recursive acronym for GNU is Not Unix) program was released under a specially created General Public Licence ("GNU GPL"). This was designed to ensure that the source-code would remain openly available to all developers. It was not intended to prevent the commercial usage or distribution of such types of software. This approach to software development was called "free software". In this context "free" meant that anyone could modify the software without the need for permission of any kind from the initial developer. However, the term "free" is often misunderstood to mean "no cost". Thus, free software doesn't mean no cost, it means open source code. Nevertheless the term "open source software" was considered less contentious and more "business-friendly", therefore it has been adopted to denote this type of software architecture.

In his essay, open source evangelist Eric S. Raymond suggests a new model for developing open software - the Bazaar model. He suggests that all software should be developed using the

bazaar model, which he described as a great babbling bazaar of differing agendas and approaches [Raymond 99].

Gregorio Robles [Robles 2004] suggests that software architecture developed using the Bazaar model should exhibit the following patterns:

- Users should be treated as co-developers. They should have access to the source code of the software. Furthermore users are encouraged to submit additions to the software, code fixes for the software, bug reports, documentation etc. Having more co-developers increases the rate at which the software evolves.
- Early Releases. The first version of the software should be released as early as possible so as to increase one's chances of finding co-developers early.
- Frequent Integration. New code should be integrated as often as possible so as to avoid the overhead of fixing a large number of bugs at the end of the project life cycle. Some open source projects have nightly builds where integration is done automatically on a daily basis.
- Several Versions. There should be at least two versions of the software. There should be a buggier version with more features and a more stable version with fewer features. The buggy version (also called the development version) is for users who want the immediate use of the latest features, and are willing to accept the risk of using code that is not yet thoroughly tested. The users can then act as co-developers, reporting bugs and providing bug fixes. The stable version offers users fewer bugs and fewer features.
- High Modularization. The general structure of the software should be modular allowing for parallel development.
- Dynamic decision making structure. There is a need for a decision making structure, whether formal or informal, that makes strategic decisions depending on changing user requirements and other factors.

Although open source software was initially applied only to computer technology, the principle of open source architecture can be applied to a variety of other applications such as robotics.

6.2.3 Open hardware architecture

Open hardware is computer and electronic hardware that is designed in the same spirit as free and open-source software. Open hardware can be considered a part of the open source culture that brings the open source ideas to fields other than software.

Some of the initiatives for the development of open source hardware began quite recently. Because the nature of hardware architecture is different from that of software architecture, and the concept of open source hardware is relatively new, no exact definition of open hardware has appeared to date.

The open software architecture definition can’t be applied directly to hardware architecture without modification because hardware has direct costs associated to it (price of processors, chips, etc.). Therefore, the term open hardware architecture was initially used to denote the use of free/open source software with hardware and the free release of information regarding the

hardware. It often includes the release of schematics, designs, sizes and other information concerning the hardware under development. The freely released information includes hardware design and element distribution on all hardware modules (central processor, motion control elements, sensors, etc).

Openness in hardware terms can have a whole range of meanings. Some hardware presented in the market is open although most is not. The current trend is for more and more limited openness of hardware architectures. This restricts the freedom of designers to create or implement their own designs with existing hardware, and even restricts the freedom of software developers to write programs that they would like to use on this hardware. Open hardware architecture would have to satisfy all the requirements below [Seaman 2004]:

- Information on using the hardware must be available.
- The design of the hardware architecture must be available.
- Designs can be free in exactly the same sense that software can be free.

As in the open software field, hardware architecture may also have different levels of openness. The following sections consider aspects of this problem.

6.2.4 Open and Standard

It can be clearly seen that open architecture should be based upon industry standards. There should be an interdependent relationship between open architecture and standards. Standards are the best methods of achieving the desired interconnection in highly modular systems. Without standards, there would be no guidelines to define such an outcome. Standardization has many benefits. When they are implemented, they permit a user access at all levels. They define common interfaces for the architecture with multiple vendors' modules.

Software architectures based on solutions considered to be industry standards implement total industry resources more effectively. Incorporating standards ensures that a product grows with the hardware and the supporting software standards upon which it is based. Furthermore, standard architectures are significantly easier to support.

It is important to highlight the difference between systems that are proprietary standards and open systems based upon a standard. For example, Microsoft Windows is clearly standard and just as clearly proprietary. Therefore Microsoft Windows is not considered by most experts to be an open operating system. Linux is clearly an open operating system, and is also a standard that operates on many different mainframes and servers. It has recently become a viable alternative standard for desktop or workstation computers as well.

Systems that are open and have open interfaces have many advantages over proprietary systems. However, many combinations that interlace open, proprietary and standard modules in the same architecture are possible. For example, it is possible to develop an open interface for a proprietary subsystem such as TCP/IP over Ethernet (open and standard) on a Windows XP (proprietary and standard) system.

Thus, while open system architectures are based on open standards and open interfaces, these systems may also interoperate with proprietary subsystems that have their own internal intellectual property rights. AMD for example, was able to reverse engineer the Intel Pentium

CPU because the Pentium interface is fully defined. However, the intellectual property associated with the manufacturing of both Intel's and AMD's CPUs remains a valuable property and its secrets are closely guarded.

Thus, it is possible for humanoid robot control architecture to use a proprietary standard Microsoft OS in a control CPU to communicate via the open standard Internet Protocol (IP) over an open wired network or an open standard IEEE-802.11 wireless network to another PC using open standard Linux OS. It is also possible for an open protocol such as internet protocol (IP) to interoperate over a proprietary wireless communications network. All these configurations can produce viable open control architectures.

However, transit properties are increasingly troubled by the high cost, rapid obsolescence and unreliability of low volume proprietary systems. Therefore, many developers now migrate away from proprietary to open systems. For this reason, many projects are beginning to successfully migrate from closed operating systems in favour of Linux OS. This migration reveals that new open paradigms systems are less expensive and more reliable.

6.2.5 Degree of openness in the control architecture

Manufacturers of controllers and robots have already begun to respond to customer demands by developing "open architecture" systems. In this context the word "open" is not synonymous with the word "universal". For example, the phrase "open controller" refers to a controller that is based on known or published specifications or standards. A "universal" controller refers to a controller that can be implemented with more than one different robotics platforms.

The degree of "openness" varies and depends on the policies of each manufacturer. One definition of open architecture is the following: "a control architecture with standard hardware and operating system with open interface specifications" [Fiedler 97]. The PC is an example of an existing open architecture system that is based on the original IBM personal computer. The PC hardware architecture now is considered a standard piece of computing hardware that is widely used in commercial products and industrial machine tools. In the same way, Microsoft Windows software can be considered a standard operating system used in millions of PCs. Therefore, the control architectures for a robot built with the implementation of the standard hardware and the standard Windows operating system has numerous advantages over closed proprietary robot control systems. But on the other hand, it is not a totally open system because it has parts that are proprietary. Therefore, in stating the openness of the control system for humanoid robots that will be presented in this section, it is necessary to study the degree of this openness.

The degree of openness in robot control architecture may vary from one system to another. Therefore the logical question is how open can the control architecture of a humanoid robot be? The robot system in the block diagram shown in Figure 6.1 illustrates one possible open control architecture system solution for a humanoid robot that was implemented in this research. For this system, some hardware components and the operating system are considered proprietary components. The motion control software algorithms and modules (described in previous chapters), the communication interface based on open architecture hardware (i.e., CAN-bus, Ethernet) and the specifications are labelled as open architecture components. The "open" label refers to the general hardware and software architecture because a proprietary but standard operating system (e.g. Windows), and standard hardware (PC, sensors) are implemented. The

quotations around the word open signify that there is a degree of openness (i.e., open relative to other systems).

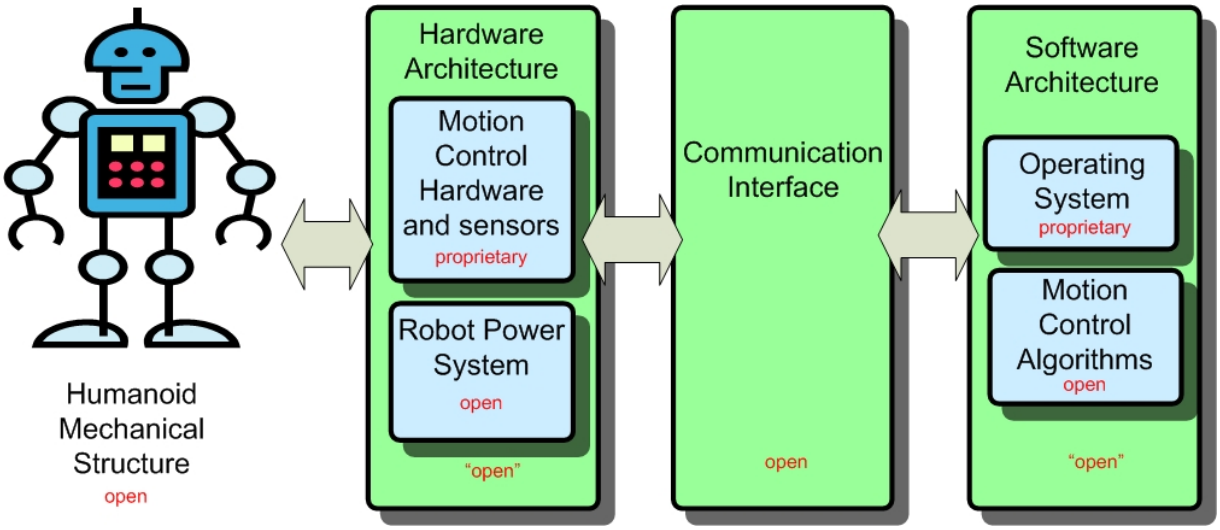


Fig. 6.1: Openness of the control architecture for humanoid robots

Therefore, the entire system can be considered as open control architecture with the use of some standard proprietary components such as industrial PC based components or the Windows operating system.

Nowadays, standard industrial PC based control architecture can more easily integrate many commercially available add-on peripherals such as; modems, Ethernet cards, mass storage devices, card scanners, sound cards and other I/O and multimedia devices. All these peripheral devices are available at reasonable consumer prices.

Developers of a standard industrial PC based control system benefit from development costs and resources invested by other companies. In other words, by using industrial PC technology in robot control architecture, developers are able to minimize costs by saving on current and future hardware devices already available in the market. PC hardware is relatively inexpensive and readily available (i.e., motherboards, CPU's, memory, etc.) and is usually produced by numerous manufacturers. In addition, this hardware doesn't have to be designed and tested individually by the developers of the control architecture for robots. As a result, the human resources and financial expenses needed to build and update a robotics system can be appreciably reduced.

The developer of an open software architecture system can benefit as much from the same wide array of available market resources as hardware developers. Software development on an industrial PC and Windows OS based system can be less expensive, faster, and more portable than with proprietary systems. Standard development tools (e.g. Visual C++, Visual Basic, Dephi, etc.) can be used. These development tools are not expensive and allow fast migration of existing source code for different CPU's and PC platforms. In addition, the number of engineers who are experienced in using these standard programming tools and the PC platform is greater than the number of engineers with in-depth knowledge of a particular microprocessor and associated compiler tools. Nevertheless, it should be taken into account that this development of

"open" architecture is based on proprietary PC hardware and proprietary OS software (Windows OS).

6.2.6 Why open architecture for humanoid robot motion control?

Proprietary (i.e., closed) humanoid robot control architecture has improved over the years, but still has many disadvantages when compared to open architecture. Most proprietary systems are viewed as "special cases" because of the "closed" nature of these biped machines and their very limited compatibility and connectivity with other systems.

Open architectures promise faster and more economical development of high-quality, technologically up-to-date humanoid robotics systems. An open systems approach is important to advancing the causes of acquisition efficiency and system interoperability [Oberndorf 97].

By using standard hardware and software components in the open architecture for humanoids, lower purchasing and operating costs and increased flexibility can be achieved. The use of standard industrial PC components in an open humanoid robot controller paves the way for many new laboratories and researchers and allows for new humanoid development and customization opportunities. The great advantage of open architecture for humanoids is that anyone can design changes and add-on products for it. By making architecture public, however, a manufacturer allows others to duplicate its design.

True, open control architecture assures modularity, interoperability across dissimilar components and systems, as well as the ability to incorporate new standards and off the shelf software tools, while remaining at an application layer that is transparent to the user. An added benefit of adopting open control architecture is that it minimizes the time it takes to develop the system and furthermore, replicates the architecture across different humanoid platforms.

The main goal of any open architecture system is to provide the user with the desired flexibility, interoperability, connectivity and reliability, while at the same time breaking the traditional reliance on a single manufacturer. With closed architecture, once a system is implemented, the robot supplier is in control. With open humanoid robot architecture, the user is in control, managing technology instead of being managed by it. In addition, open control architecture not only makes the user become familiar with current control systems technology, but compels them to establish future technology requirements. This is an important attribute of flexibility that open control architecture for humanoid robots can provide.

6.3 Hardware architecture design

6.3.1 Considerations on hardware architecture design for humanoid robots

The word hardware is an expression used within the engineering disciplines to explicitly distinguish the (electronic computer) hardware from the software which runs in it.

Hardware architecture is a representation of an engineered electronic or electromechanical hardware system, and the process and discipline for effectively implementing the design(s) for such a system [Frampton 2003]. It is generally part of a larger system encompassing information and software. It is a representation because it is used to convey information about the related elements comprising a hardware system, the relationships among those elements, and the rules

governing those relationships. It is a process because a sequence of steps is prescribed to produce or change the architecture, and/or design from that architecture a hardware system within a set of constraints.

Hardware architecture is primarily concerned with the internal electrical (and, more rarely, mechanical) interfaces among the system's components or subsystems, and the interface between the system and its external environment, especially the devices operated by or the electronic displays viewed by a user. (This latter, special interface, is known as the computer human interface, or human computer interface, or HCI; formerly called the man-machine interface.) [Brunelli 2008]. Thus, as mentioned above, hardware architecture is an abstract representation of an electronic and/or electromechanical device which is capable of running a fixed or changeable program [Assif 98].

In the case of a robotics system, hardware architecture is the physical part of a robot, including the digital circuitry, as distinguished from the robot software system that executes within the hardware. The hardware of a robot is infrequently changed, in comparison with software which is "soft" in the sense that it can be readily created, modified or erased on the robot.

Explicit restrictions on the architecture for a humanoid robot are posed by the limited availability of space. Since the robot should operate autonomously, the problem of energy consumption arises. The robot has to carry all its energy resources with it. As electrical energy can not be stored efficiently, care must be taken to choose components with low energy consumption.

Viewed abstractly a robot is a data processing machine. It consists of data sources, a data processing device and data sinks [Regenstein 2003]. Data sources are mainly a variety of different kinds of sensorial systems. There are, for example, optical encoders, force torque sensors, microphones, cameras or inertial sensors. This list shows that on the sensory side there are extremely heterogeneous inputs that must be taken into account when designing hardware architecture for humanoid robots.

Another aspect that shows how different the sensors requirements are can be seen in the bandwidth and cycle times required. For example, cameras demand high bandwidth but their cycle time may be in the area of approximately 20 ms. Compare to this the bandwidth requirements of actuator or stabilization control which are much lower but whose cycle time must be as low as 1 ms.

Data processing is usually done in some kind of PC or microcontroller system. Looking at the topology of a robot i.e. the distribution of available space for data processing, it is obvious that in most cases data sources and data processing cannot be located in the same place. For example force/torque sensors should be placed in the foot of a humanoid robot, but logically, there is no possibility of processing this data there. Usually the greatest available space in a humanoid robot is in the torso. However, most of the data sources e.g. cameras and sensors for motion control are in the extremities of the robot. A suggestion for hardware architecture that is able to deal with these diverse sensors is presented in this section.

As for data sinks, there is the data processing aspect itself and the actuators of the robot that are heterogeneous as well. Although electric motors are usually used for locomotion in humanoid robotics, there are also different approaches, such as pneumatic or hydraulic actuation. And even among electric motors, there are different types such as brushed or brushless motors.

To summarize all the requirements for the architecture posed in chapter 3 and previous sections of this chapter, hardware architecture for humanoid robot motion control should be:

- open
- hierarchical
- modular
- distributable
- scalable
- be able to allow the use and implementation of standard interfaces

Especially in humanoid robots which are autonomous systems and restricted by the limited space available inside the robot's body, there are additional requirements such as:

- energy efficiency
- compact size
- lightweight
- ease of cabling

The detailed design of this hardware architecture is presented below.

6.3.2 Basic systems of hardware control architecture

The control architecture of a humanoid robot implies the organization of the control system. In general, there is a global humanoid robot motion control application divided into different subtasks. As in software, where each subtask is performed by a special subprogram, in hardware, control is carried out by special hardware systems which can be made up of sub-systems. Analysing a humanoid robot control task makes it possible to highlight the following principal hardware systems:

- Main Control
- Joint Control
- Communications
- Sensors
- Head system
- Power control electronics
- HMI control PC

Figure 6.2 shows the structure of a humanoid robot with the principal hardware systems onboard.

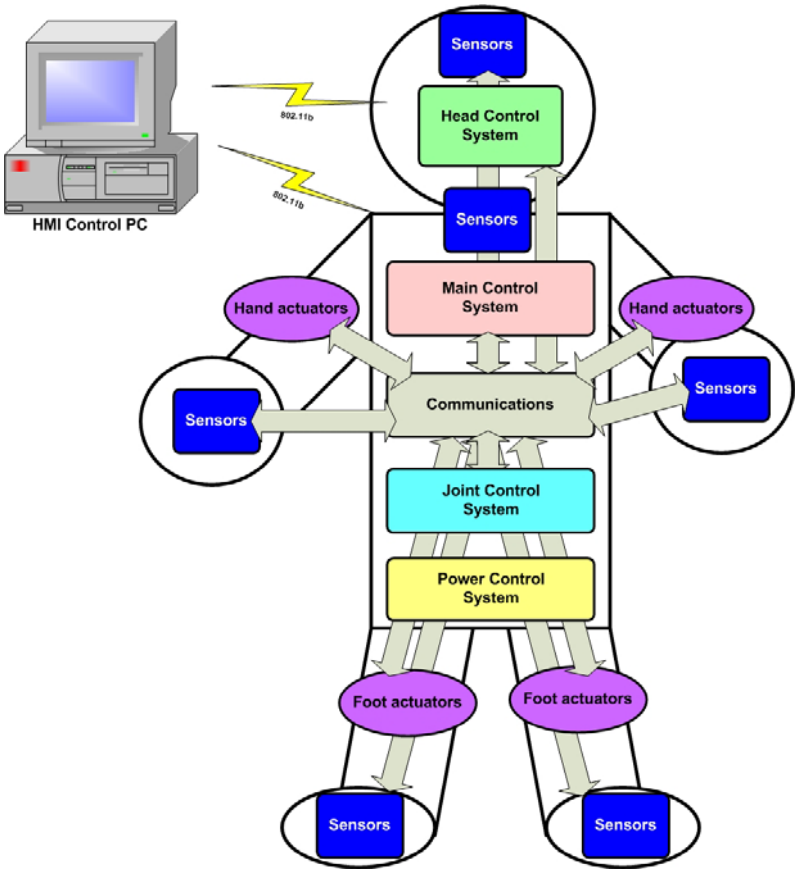


Fig. 6.2: Basic systems of a hardware control architecture

As can be observed in figure 6.2, the principal hardware systems are physically located in different parts of a mechanical structure and are interconnected to each other by a communication system. All basic hardware systems of a humanoid robot have their own specific purpose and should be carefully designed and mounted.

The *Main control system* of a humanoid robot performs the main data processing and computing tasks. It should execute upper level control algorithms (such as stabilizer control) and generates table motion patterns for each joint of a robot. Also, it performs general synchronization of all hardwire onboard.

The *Joint control system* provides synchronous multi-axis motion of every joint and performs bottom level control tasks such as trajectory execution control.

The *Communication system* should provide reliable connections among all the systems of a robot. Also it communicates the onboard control system with the external computer and the human - user. The *Sensorial system* provides data to be used in motion control process. It transmits the real world and the robot's current states to the control system of a robot.

The *Head control system* is responsible for robot's interaction with the environment. Usually it is used for sound and image processing. It doesn't have a direct influence on the motion control system of a humanoid and, therefore, will not be considered in further detail within this work.

The *Power control system* provides power to all elements of hardware architecture.

The *HMI Control PC* is an external computer providing the HMI for the user.

All these hardware parts are critical for the motion control architecture of a humanoid and will be considered in detail in the following sections.

6.3.3 Hierarchy of the hardware architecture. Levels of control

One of the requirements for the control architecture of a humanoid robot posed in chapter 3 was the hierarchical structure of its components. If we take a look at the componential base of hardware for a humanoid robotics system, we find that the componential base, consisting of sensors, actuator, etc. is practically the same as that in the industrial automation field. Contemporary industrial automation systems can be very complex, and are usually structured into several hierarchical levels. Therefore, as in the industrial field, the motion control architecture for a humanoid robot can be divided into several levels of hierarchy (Figure 6.3).

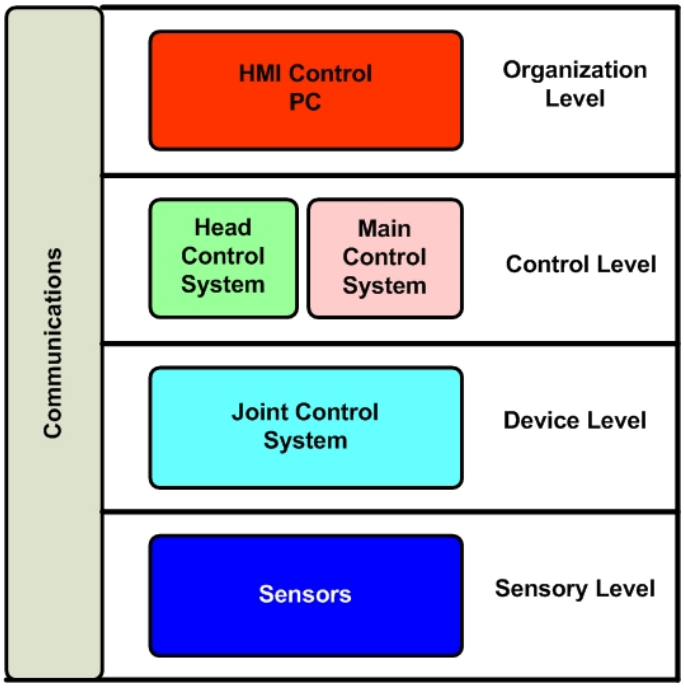


Fig. 6.3: Hierarchy of hardware architecture

The lowest level of the hierarchy is the *sensory leve*l which includes the sensors needed for carrying out humanoid robot stable bipedal locomotion. The task of this level is to acquire and transfer data about the environment and the robot's current state. The data may be either binary or analogue. Measured values may be available for a short period of time or over a long period of time.

The *Device level* contains the joint control system of a humanoid robot. At this level, the information flow mainly consists of loading motion control programs, parameters and data. The

control processes of this level require devices with short machine idle times and readjustments. This determines the timing requirements.

The *Control level* consists of the Main (composed of different controllers) and Head control systems. The control level is designed to be application-oriented. Among the controlling and intervening functions at this level are activities such as the setting of control targets and emergency activities.

The *Organization level* is the top level of a of a humanoid robot control system. The organization level PC gathers the management information from the bottom levels, and manages the whole control system.

Communications provide the data flow and interaction between all levels of a control system.

6.3.4 Main control system

6.3.4.1 Requirements

The main control system is the “brain” or core of a control architecture, which controls the motion of a humanoid robot. Its basic task is to produce different kind computations. Computations are produced by computers, thus, the main control system is a computer with extended computational and communicational capabilities. Its most important functions are:

- Sensorial data processing
- Generation of a stable motion trajectories
- Synchronization of all joint motions
- Stabilization control of a robot
- Data exchange with other modules of a control system

In the proposed architecture, in order to fulfil the criteria of autonomous performance, the main control system should be mounted inside the humanoid robot’s body. The increase in weight of a system increases the forces and torques needed to move the robot. Therefore, the computer used in the control system of a humanoid should be lightweight and have reduced power consumption. These requirements make it impossible to use a standard personal computer in the control architecture. One possible solution to the problem of implementing computational and control tasks given such restrictions is the use of embedded computer systems.

6.3.4.2 Embedded systems

An embedded system is a special-purpose computer system designed to perform one or several dedicated functions, often with real-time computing constraints [Barr 2007]. Generally speaking, an embedded system is a computer created for a special purpose. It can be embedded as part of a complete device (humanoid robot) including hardware and mechanical parts. A general-purpose computer, such as a personal computer, can do many different tasks depending on what programs it has. An embedded system is based on specifications posed by a designer and is concentrated on a few previously defined tasks. Specific requirements should be carefully defined and an

embedded system should be developed based on these. As an embedded system works on a predefined task, it increases the productivity of the whole control system.

Embedded systems are usually developed to perform specific tasks in order to lower costs or include other characteristics that general-purpose computers lack. Lowering the cost reduces the speed of an embedded system. But most functions associated with embedded systems do not require very high speed. Most often speed is not an issue and one achieves the task at lower cost. Simplifying the hardware allows cost reduction. Other characteristics that are usually included in embedded computer systems are related to the overall dimension restrictions in the final product.

Basic characteristics of embedded systems which comply with restrictions posed by control architecture of humanoid robots and thus, make the implementation of such systems preferable, are:

- Embedded systems are designed to do some specific task that a general-purpose computer is not able to do or does but in a less cost effective manner. Depending on the application, some embedded systems have real-time performance constraints that must be met. Other systems may have low or no performance requirements, allowing the system hardware to be simplified to reduce costs.
- Embedded systems are usually not separate devices. They can be easily physically built-in to the devices they control. This is one of the advantages of this kind of system over standard general purpose computer systems.
- The software modules suitable for embedded systems are often called firmware. These modules are stored in the memory of the device or on a disk drive. Often it is able to run with very limited hardware resources: small or no keyboard, no screen, and little memory.
- As the embedded system is dedicated to always performing previously specified tasks, designers can optimize it, reducing the size and cost of the final system, or increasing the reliability and performance if needed for the application. On occasion, the economy of scale of mass produced embedded systems benefit the developer.

6.3.4.3 CPU platform

The core of a main control system is the central processing unit (CPU) module. It carries the main computational load of a system. The CPU is always used to process the data, execute any motion control program and send an order to other control systems. In the case of its implementation in robotics system, a CPU module has to meet the following requirements:

- It should have sufficient processor speed for its average computational load and good terminal characteristics. In another words, the rule “faster is better” does not apply. A very powerful processor consumes a lot of electrical power and emits heat that may lead to overheating in the closed robot body or need a very complex ventilation system. On the other hand, a humanoid robot motion control system doesn’t need very fast data processing because there are too many other decelerating factors such as a bus bandwidth which will be considered later.

- It should have a data interface for data transmission from different sensorial systems. Most sensors have RS232 or USB interface. Therefore, the CPU module should have enough of these ports.
- It should have a networking interface for data exchange with the other modules of a control system, for example sound and image processing systems.
- It should be easily compatible with most operating systems and programming environments.
- It should be small and light weight.

Embedded systems can be broken into two broad categories according to type: ordinary microprocessors and microcontrollers, which have many more peripherals on chip, reducing cost and size. In contrast with the personal computer and server markets, a fairly large number of basic CPU architectures with embedded systems are used. Word lengths vary from 4-bit to 64-bits and beyond (mainly in DSP processors) although the most typical remain 8/16-bit. Most architectures are available in a wide range of models and configurations, many of which are even manufactured by several different companies. A long but not exhaustive list of common architectures is: 65816, 8051, ARM, AVR, Blackfin, Coldfire, COP8, HT48, M16C, M32C, MIPS, MSP430, PIC, PowerPC, SHARC, Tricore, x86, etc.

In analyzing the variety of CPU architectures, among the first issues that should betaken into account are not only the functional but also the conceptual requirements. As mentioned above, the motion control architecture of a humanoid robot should meet the requirements of standardized interfaces and modules. This eliminates the vast majority of CPU architectures that might be implemented in the main control system because most of these architectures don’t meet these criteria. After careful study, it has been concluded that the most appropriate CPU architecture is the standard x86 architecture.

The generic term x86 refers to the instruction set of the most commercially successful CPU architecture in the history of computing. It is used in Intel, AMD, VIA, and other processors. The term x86 is derived from the model numbers of the first few generations of processors, that were backward compatible with Intel’s original 16-bit 8086 CPU. Since then, many additions and extensions have been added to the x86 instruction set, almost always with full backwards compatibility.

The main advantage of all x86 based CPU architectures is that they can be implemented with a variety of standard buses, IO modules, peripheral devices and software systems. It is possible to use all the experience gained in the personal computer field (in addition to personal computers, this architecture is implemented in most industrial computers) to create an effective motion control architecture for humanoid robots. It allows developing time and costs to be appreciably reduced.

6.3.4.4 Form factor

A computer system form factor is the parameter that defines the different sizes and de-facto standards of the motherboard used with different x86 processors. Differences between form factors are most apparent in terms of their intended market sector (personal or industrial computer), and involve variations in size, design compromises and typical features. As

mentioned above, a standard personal computer doesn't meet the requirements posed for humanoid robot motion control architecture hardware. Thus, the most obvious choice is the industrial form factor for x86 CPU architectures. Current tendencies in the industrial computer market indicate that one the most popular form factor standards is the PC/104 form factor computer. Appendix A presents the description of the PC/104 computer standard.

To conclude, it is necessary to mention that the wide use of PC/104 form factor computers is caused by PC based bus architecture. By standardizing hardware and software around the broadly supported PC system, humanoid robot embedded control architecture designers can substantially reduce development costs, risks, and time. Another important advantage of using PC architecture is that its standardized hardware and software components are widely available. These components are also significantly more economical than traditional non-PC bus architectures such as STD, VME, or Multibus.

6.3.5 Joint control system

6.3.5.1 General scheme

The main objective of a joint control system is to provide a humanoid robot with joint motion. As mentioned in chapter 3, all robot joints are usually positioned (controlled) in a closed-loop. Simple closed-loop control has been used for decades to perform machine tool contouring. But the extreme complexity of the applications in humanoid robotics aimed to handle special control applications that may include synchronization of multiple axes. These applications require additional improvements to handle changing loads and inertias when positioning the joints of a humanoid robot.

In general, the joint control system of a humanoid robot should be able to generate set points (the desired output or motion profile) that position a joint in space, produce the motion and close a position and/or velocity feedback loop. Therefore, the basic architecture of a joint control system should contain:

- A driver or power amplifier to transform the control signal from the motion controller to a higher power electrical current or voltage that will then be transmitted to the actuator. New "intelligent" drivers can act as the motion controller closing the position and velocity loops internally.
- An actuator. In most robotic systems the motion is produced by an electrical servo motor.
- Feedback sensors such as optical encoders, resolvers or Hall effect devices. These sensors are used to read the actual position and/or velocity of the actuator and send it to the motion controller in order to close the position and/or velocity control loops.
- Mechanical components used to transform the motion from the actuator to the other elements of the system. These components include gears, shafting, ball screw, belts, linkages, and linear and rotational bearings.

In a basic joint control system, Figure 6.4, the load represents the mechanical link being positioned. The load is coupled or connected through one of the mechanical linkages. The motor may be a traditional DC servo motor, a vector motor, or a brushless servo motor. All motor

functions such as starting, stopping command, speed, reference position, etc. are dictated by the motion controller. The amplifier is the part of the motion controller that takes a low level incoming command signals (+/- 10V) and amplifies it to a higher power level to drive the motor.

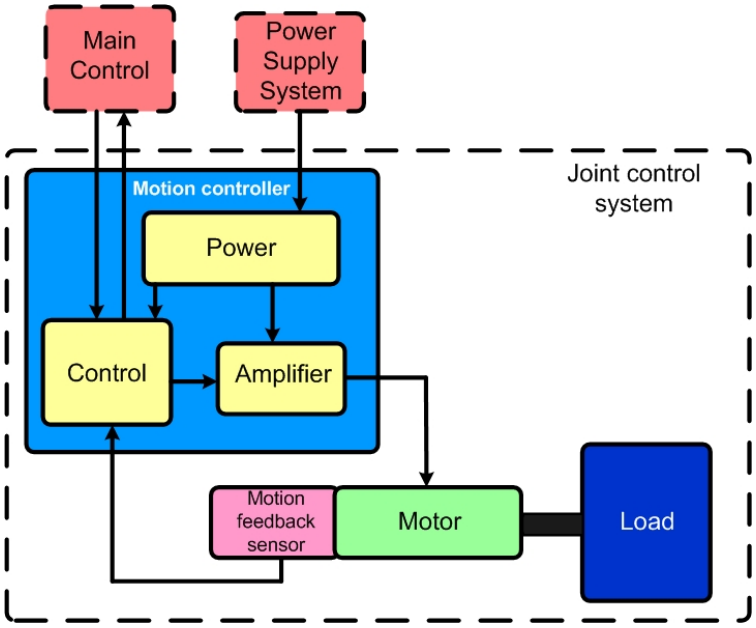


Fig. 6.4: Joint control system

The programmable motion controller is the brain of the motion system. The motion controller is programmed to accomplish a specific task for a given application. It receives upper level orders from the main control system and provides the motion of a motor. It reads a feedback signal to monitor the position of the load. By comparing a pre-programmed, “desired” position with the feedback “actual” position, the controller can take action to minimize any error between the actual and desired load positions. In addition, contemporary motion controllers are provided with amplifying systems which allow a +/- 10V control signal to be sent to the motor and provide the needed level of a control current.

6.3.5.2 Motor selection

An electric motor is a device that uses electrical energy to produce mechanical energy. Usually electric motors are divided into DC types and AC types. It is clear that AC type motors can’t be implemented in autonomous systems such as a humanoid robots because they require an AC power supply. Therefore, the most appropriate choice is a classic DC motor solution.

A DC motor is designed to run on DC electric power. All DC motors can be divided into 3 main groups by how they transform electrical energy into rotary torque [Gottlieb 94]:

- Brushed DC motors. The classic DC motor design generates an oscillating current in a wound rotor with a split ring commutator, and either a wound or permanent magnet stator. A rotor consists of a coil wound around a rotor which is then powered by any type of battery. Many of the limitations of the classic commutator DC motor are due to

the need for brushes to press against the commutator. This creates friction. At higher speeds, brushes have increased difficulty in maintaining contact. Brushes may bounce off the irregularities on the commutator surface, creating sparks. This limits the maximum speed of the machine. The current density per unit area of the brushes limits the output of the motor. Imperfect electric contact also causes electrical noise. Brushes eventually wear out and require replacement, and the commutator itself is subject to wear and maintenance. The commutator assembly on a large machine is a costly element, requiring precision assembly of many parts.

- Brushless DC motors. Some of the problems of the brushed DC motor are eliminated in the brushless design. In this motor, the mechanical "rotating switch" or commutator/brushgear assembly is replaced by an external electronic switch synchronised to the rotor's position. Brushless motors are typically 85-90% efficient, whereas DC motors with brushgear are typically 75-80% efficient. Brushless DC motors are commonly used where precise speed control is needed.
- Coreless DC motors. Nothing in the design of any of the motors described above requires that the iron (steel) portions of the rotor actually rotate; torque is exerted only on the windings of the electromagnets The coreless DC motor, a specialized form of brush or brushless DC motor, takes advantage of this fact. Optimized for rapid acceleration, these motors have a rotor that is constructed without an iron core. The rotor can take the form of a winding-filled cylinder inside the stator magnets, a basket surrounding the stator magnets, or a flat pancake (possibly formed on a printed wiring board) running between upper and lower stator magnets. The windings are typically stabilized by being impregnated with epoxy resins. Because the rotor is much lighter in weight (mass) than a conventional rotor formed from copper windings on steel laminations, the rotor can accelerate much more rapidly, often achieving a mechanical time constant under 1 ms.

The choice of which electrical motor (and motor type) to implement in the motion control depends on the dynamical characteristics that comprise the entire system. Once the mechanics of a humanoid robot have been analyzed, the friction and inertia of the load of each joint are known, the next step is to determine the torque levels required. Then, a motor can be sized to deliver the required torque and the control sized to power the motor. If friction and inertia are not properly determined, the joint control system will either take too long to position the load, burn out, or be unnecessarily costly. If the real required torque level in some parts of a joint trajectory is higher than the maximum torque generated by a motor, the joint will not follow the desired motion pattern exactly and a humanoid robot can lose its balance and fall down. Thus, the selection of appropriate motors for each joint is a very important design task that requires a complex dynamic study of the system.

6.3.5.3 Motion controller selection. Distributed vs. Centralized joint control

In the case of multi-axis operations (when the architecture consists of more than one motion controller) it is necessary to consider how to control all participating joints optimally. Moreover, as mentioned above, a humanoid robot system poses very strict restrictions on the size and physical distribution of hardware control devices.

Currently, all multi-axis joint control methods can be divided into two basic groups - distributed or centralized architectures. A “classic'” centralized motion control architecture is presented in figure 6.5.

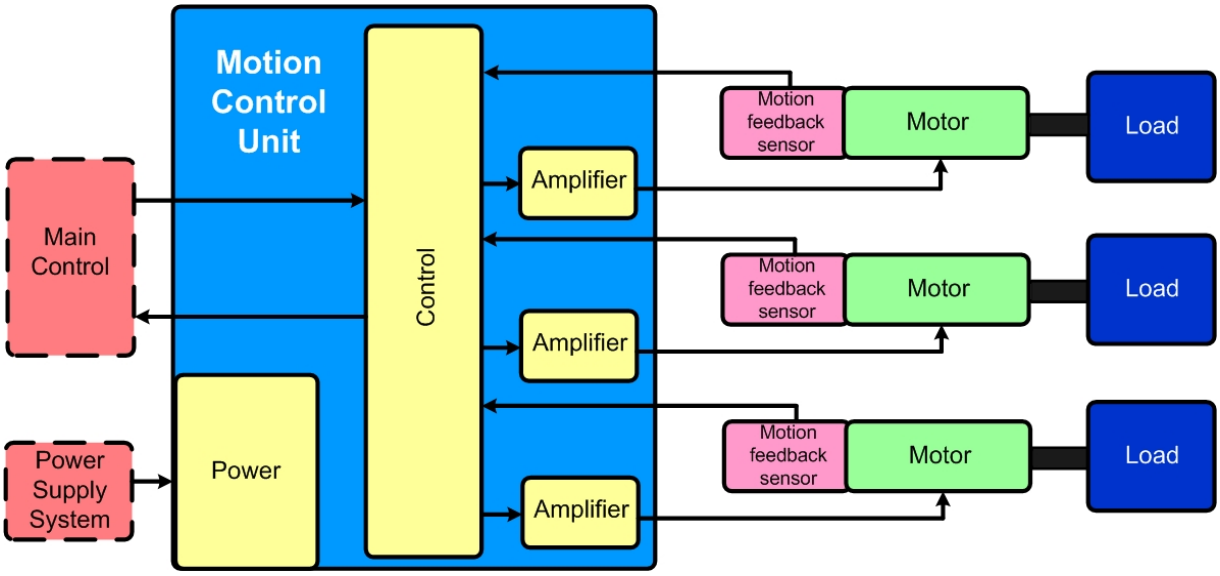


Fig. 6.5: Centralized joint control

The centralized approach commonly uses a multi-axis servo motor control unit (or controller) that outputs ±10-V analogue control signals or pulse-and-direction signals. These control signals serve as inputs to amplifiers that in turn output higher power signals to the motor. Feedback signals from each motor provide the encoder position to the controller or, in some cases, the controller and the amplifier.

A distributed architecture is presented in Figure 6.6. The distributed approach gives more intelligence to the amplifier [Lewin 2001]. The main controller has no motion control card; instead it sends commands over a network bus to the intelligent motion controllers that receive and process the commands. These intelligent devices then close the servo loop, and typically perform some trajectory generation as well.

To benefit from less wiring, the amplifiers should be located near the motors. In the latest modern distributed control models, the amplifier and motor are combined into a single ultra-integrated unit completely eliminating any external wiring between the motor and the amplifier. This type of unit is sometimes called a “smart” motor or integrated motor. As can be clearly seen in figure 6.6, distributed joint control uses a network bus to reliably connect the Main Control module with the intelligent motion controllers and other components residing on the network. With either architecture, additional motion control functions can be executed in the main control module.

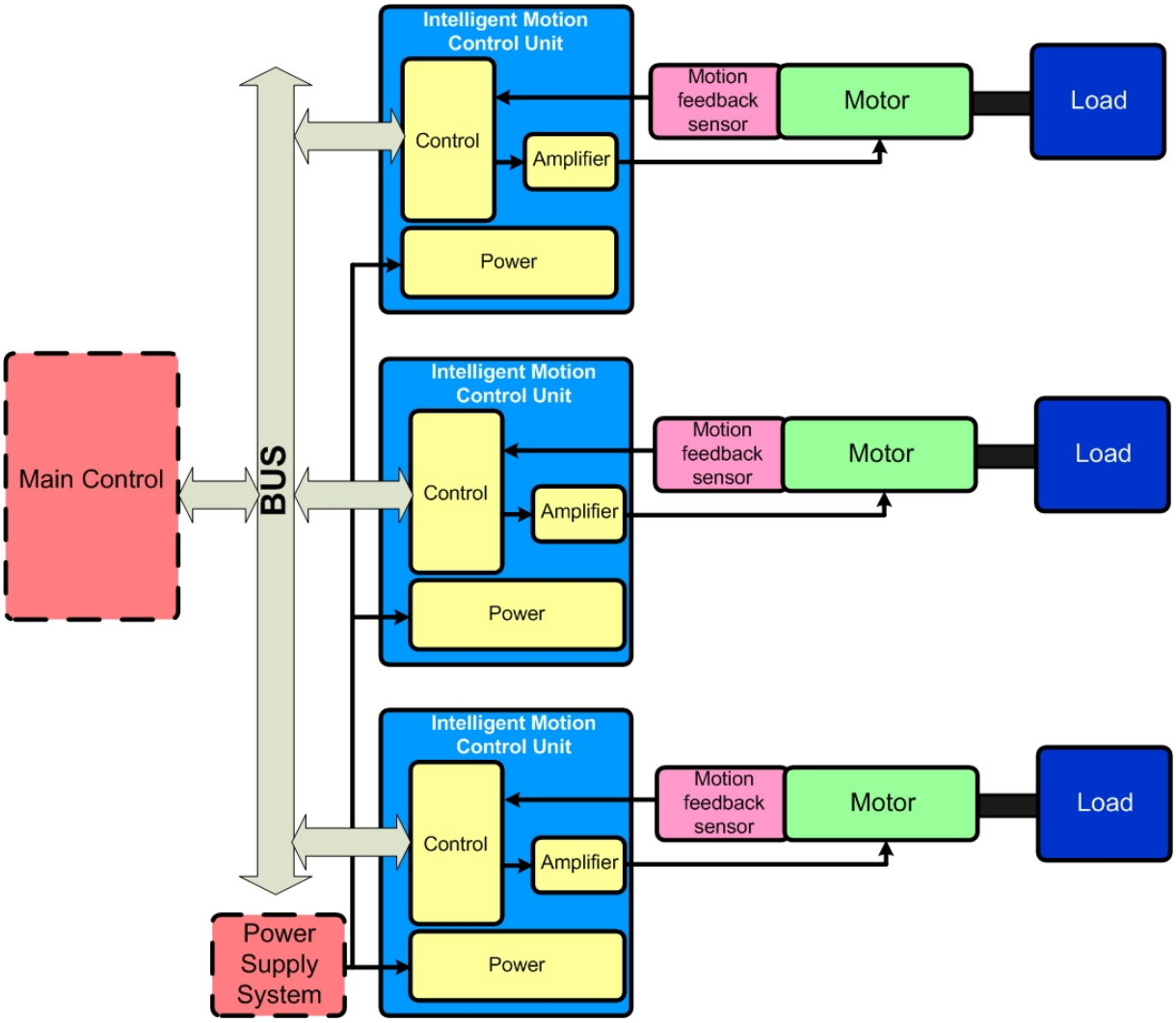


Fig. 6.6: Distributed joint control

Summarizing, the choice of multi-axis distributed control for a humanoid robot architecture design is based on three basic factors:

- *Mechanical configuration* of a humanoid system. The distributed approach should, in theory, result in higher reliability due to reduced wiring. Moreover, it also reduces the size of control components and allows them to be distributed more effectively inside the robot's body. It requires very careful consideration of the mechanical and kinematic organization of a humanoid robot.
- *Sensor interactivity*. The assembly of motion control architecture for humanoid robot requires the implementation of different sensorial systems in addition to motors. Thus, the behaviour of the entire humanoid's motion depends on the status of signals from another part of the control system (for example force-torque or inertial sensors). This leads to the need to include sensors in the entire control process and place the system under distributed architecture for signal exchange using the same bus.

- *General architecture*- This refers to the native organization of the control system. This distributed philosophy exactly meets the requirements for modular hierarchical distributed architecture for humanoid robots proposed in chapter III. It is modular and distributed because the control of each joint is executed in a special module – intelligent motion controller. It is hierarchical because the main control module has higher priority, in another words it is a master and the intelligent motion controllers act as slaves.

6.3.5.4 Joint synchronization in a complex multi-axis interaction

A distributed multi-axis joint control system usually comprises a multicast communications network having several node components – intelligent motion controllers, sensors, etc. Each of the intelligent motion controllers includes a clock and an actuator. The actuators are part of the general control system of a humanoid robot and generate a pattern profile table for stable bipedal motion of the entire system. The general motion pattern profile table is translated into a single-direction-of-motion pattern table and sent to each node component of the multi-axis system. The motion of each actuator is driven by its own motion controller separately from other actuators. A grandmaster (main control system) clock generates synchronization signals which are transmitted through the network at a sync interval and synchronize the clocks of each device residing on the bus. Time-stamps are generated at an interval which is a whole number multiple of the sync interval. The time-stamps cause concurrent execution of the first and next steps from the single-direction-of-motion pattern tables to produce synchronized multi-axis motion of the entire humanoid robot.

6.3.6 Communication system

6.3.6.1 Requirements

Once we have considered the design issues outlined above, it is necessary to select the appropriate network bus that will provide the communication between all components in distributed motion control architecture.

Hardware architecture is characterized by its components and the connections between these components. First of all, the connection between the components must have sufficient bandwidth. As the whole motion control architecture has to be able to process data in real time, the connection must be able to transmit data in real time as well. Other requirements are: there should be means to recognize an error in the transmission and the number of supported nodes should be sufficient. For use in a humanoid robot system, some other aspects are especially important. These are, for example, ease of cabling. This is important because a thick cable harness would constrict the movement of humanoid robot's multiple joints. Ideally there would only be a pair of wires for power supply and another pair to command the actuators. Contemporary industrial networks meet all these conditions. However, the variety of motion control applications has lead to the proliferation of communication networks. This problem will be considered in detail later in this section.

The data exchange between a humanoid robot (its motion control system) and its external control station, operated by a human, makes it necessary to implement other communicational

technologies, but the basic requirements remain practically the same. These communicational systems should provide adequate bandwidth. In addition, it is necessary to include wireless data transmission capacity for humanoid robot autonomy. Real-time transmission in such systems is not a compulsory condition because upper level control processes (sending global motion commands such as “start the motion”, “lift left hand”, etc. or motion data visualization) are not real-time and depend on human operator reactions.

6.3.6.2 Proliferation of communication standards

Basically, in distributed motion control architecture, an intelligent motion controller receives signals from sensors or the main control system, processes them using defined control laws, and generates the required output signals to drive an actuator. This raises the question of what communication standard should be implemented with a motion control system? Each of the existing elements of the motion controller architecture has specific standards. However, it is in the area of communication buses that the proliferation of standards has occurred as a consequence of the breadth of the motion control domain.

This has occurred because motion control in general is a broad domain that spans a variety of applications, includes a wide range of technology, and is comprised of many different functions. Different application demands range from simple position control of a single axis, through coordinated control of multiple axes, along with cross coupling, dynamic compensation, and integration with process parameters. Motion control technology in general encompasses sensors for proximity, position, velocity, acceleration, force, and temperature, and actuators such as electric rotary and linear motors, hydraulic or pneumatic pistons, piezoelectric or magnetostrictive actuators, and micro-electro-mechanical system (MEMS) devices. Motion control functionality ranges from control of the moving device itself using current or other motive forces at the microsecond time scale, through high-level programmed motion with time scales of minutes or longer, and all the stages in between. These numerous dimensions of motion control make standardization of communication buses and as a result, adequate selection for implementation into an open motion control system for humanoid robots a complex task. The actual tendency to focus on particular application areas has led to a proliferation of communication standards that are all related yet all different.

This would not present a problem if each standard covered a different part of motion control functionality, but this is not a case. The available communication standards cover a wide range of capability and cost ranging from high-speed networked IO subsystem standards to distributed communications standards for integrating all hardware components into a complex system. Numerous competing communication standards are available to interface the intelligent motion controller to the sensors and main control modules. Furthermore, application domain communication standards compete with other domain communication standards as well as with PC industry communications standards. In fact, many PC communications standards have been adapted to motion control to provide built-in redundancy and durability, so that all the devices remain connected despite the harsh conditions often found in a motion control environment.

All existing industrial communicational standards can be reviewed based on four levels of communication: organization, control, device and bit or sensory level [Proctor 2003]. Figure 6.7 summarizes communication performance by level of automation in the industrial system. Basic characteristics of the communication standards are presented in Appendix B.

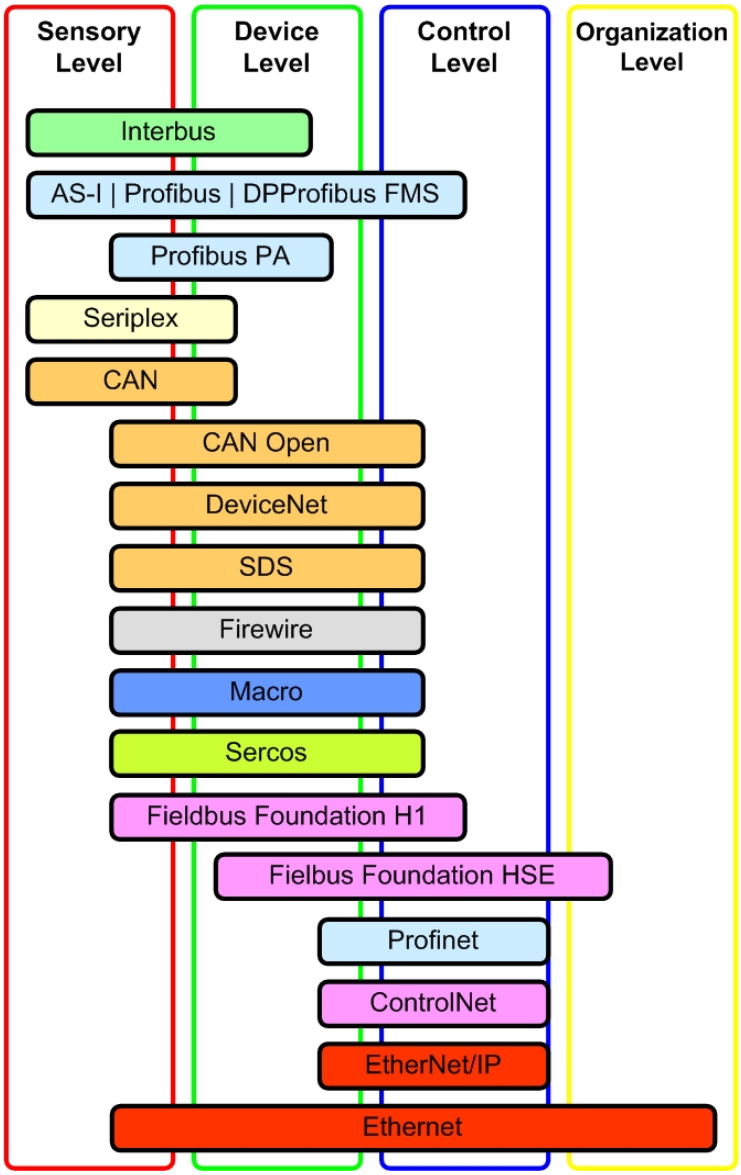


Fig. 6.7: Communication standards within the realm of the industrial automation communication hierarchy [Proctor 2003]

Organization level communication is responsible for connecting the external HMI control computer with the motion control system inside the humanoid robot. This communication can be made by using either Local Area Networks (LAN) or Wireless Local Area Network (WLAN) technologies. Organization level communication provides critical upper level information for robot motion, for creating a robot state parameters database, and for providing links between user and robot. It usually transmits big packages of control data for monitoring all control processes.

Control level communication can be based on LAN networks that provide broadcast and point-to-point messaging between nodes, and perhaps to other networks via bridges. Control level

networks are used for data gathering, interprocess exchange of control data and sequencing information, as well as remote access.

Device level communication provides cost-effective cabling and connectivity to intelligent devices (motion controllers) that control the motion of humanoid robot joints. This communication level requires hard real-time data transmission.

Sensory level communication deals with the lowest level of automation and provides low-cost networks for connecting binary devices, such as sensors and relays, with higher-level control devices. Communication solutions at this level are often called sensor-actuator busses with deterministic, real-time protocols to transmit control and device data, configure the system architecture, power the devices, and monitor the network.

Table 6.1 summarizes communication performance of communication systems by level.

	Type of data for transmission	Network Size	Data Size	Response Time
Organization Level	Upper level command and data	Large >100 nodes	Large	< seconds
Control Level	General control data	Moderate < 100 nodes	Medium	<10 msec
Device Level	Motion control data	Small < 64 nodes	Small	1-10 msec
Sensory Level	Sensorial data	Small < 64 nodes	Small	<1 msec

Table 6.1: Communication performance of communication systems

The superposition of the basic existing standards in every level of automation that makes choosing the best solution to be adopted for the hardware architecture of humanoid robots a very difficult task is shown in figure 6.7. Moreover, it indicates that Ethernet based networks are the preferred choice for motion control architecture for humanoid robots. Because of the low cost of Ethernet based products, there has been a long-standing desire to use Ethernet as a “backbone” to connect the sensory, device, control, and organization levels.

Although highly desirable, the use of Ethernet in all level of control architecture is unrealistic as Ethernet is non-deterministic due to its bus arbitration scheme. Furthermore, Ethernet connectors present a problem in applications where vibration is present.

Nevertheless, after analysing all the new trends in contemporary control engineering it can be stated that the best solution for humanoid robot control architecture is the implementation of CAN bus based networks in the time-dependant Sensory and Device levels and LAN/WLAN Ethernet based networks in the Control and Organization levels of the architecture where strict real-time is not needed. Further argumentation of this choice and basic characteristics of CAN bus and LAN/WLAN technologies can be found in Appendix B. All other communication aspects of the control system will be considered later in the section on communication infrastructure design.

6.3.7 Sensorial system

6.3.7.1 Requirements

The sensorial system of a humanoid robot is a data acquisition system. It is a system designed to measure and log certain control parameters. The data acquired by the sensorial system is used to analyse the environment around the robot and its current state in order to improve (by adjusting to a target value) measured characteristics. The data acquisition system is normally electronics based, and consists of hardware and software parts. The hardware part is made of sensors for sensing the environment, cables for transmitting the information and electronics components for converting measured signals into the desired form.

A sensor is a device that measures a physical quantity and converts it into a signal which can then be read by an observer or by another control device [Kretschmar 2005]. It is known that an important relationship exists between the performance of the control system and the characteristics of the sensorial system. The basic characteristics of a sensor are sensitivity and resolution. A sensor's sensitivity indicates how much the sensor's output changes when the measured quantity changes. The resolution of a sensor is the smallest change it can detect in the quantity that it is measuring.

The main requirements for the sensors implemented into the sensorial system for humanoid robot are:

- Sensitivity to the measured properties
- Insensitivity to any other property
- No influence on measured property

As shown in chapter 5, the sensorial system for humanoid robot motion control (besides position sensors like encoders) should comprise force/torque and inertial (gyroscope and accelerometer) sensors.

6.3.7.2 Force/Torque sensor

A force-torque measuring system in humanoid robotics is used for obtaining actual forces and torques acting one the foot in its contact point with a ground. It consists of a transducer mounted at the contact point of a robot end effectors with the environment, connected by a flexible cable or computer bus to a sensor interface controller [Perry 2002]. The transducer converts obtained force and torque, loads it into strain gage signals and transmits them to the controller of the sensor. The sensor controller performs computations on received data and provides force and torques vectors to the robot's motion control system.

Transducers calculate expected moments and forces. A transducer is usually sized by the moment capacity of the system. A humanoid robot end-effector attached to a transducer generates forces when performing a task (stepping over the ground). The measured moment can be computed by multiplying the applied force by the distance from the transducer's origin to the point at which the force acts.

When designing the force-torque sensorial system it is important to consider all possible overload conditions as well as normal operating forces and moments in order to avoid damage to

the sensor. This encompasses all loads the transducer sees including those the application doesn't monitor. During its walking motion, a humanoid robot can handle and create much larger loads, but with some loss of positional repeatability. Moreover, sensorial systems are typically overpowered for an application and are capable of exerting loads many times the rated capacity.

The prevention of damage to the sensorial system from overload depends mostly on the type of strain gages used. In order to withstand higher overloads, the sensorial system should use transducers with high-output strain gages. High-output strain gages also decrease the level of noise because less signal amplification is needed. For example, silicon strain gages provide a signal 75 times stronger than conventional metal foil gages.

The next step in the designing of a force/torque control system should be the identification of transducer capacity. Transducers should be selected and calibrated based on minimum and maximum orthogonal forces (Fx, Fy, Fz) and torques (Tx, Ty, Tz) to be measured, weight, and physical size of the sensor. Manufacturers of sensors typically provide designers with special tables to cross reference measurement ranges with available transducer types.

The next consideration should be the resolution and accuracy of the system. Logically, transducers with finer resolutions will have lower moment capacity and vice-versa. In all cases, transducer output resolution is much finer than absolute accuracy so it is necessary to ensure that the absolute accuracy level fits the application.

Sensor controllers receive information from the transducer and compute resolved force and torque data. After that, onboard control software should multiply strain gage vectors by a calibration matrix to form three orthogonal forces and torques. The force and torque data is transmitted to the main control system of a humanoid robot as control signals. Most commercially available sensor controllers can output six load axes, do tool transformations that move the center-of-origin to a user-specified location, and detect and store peak F/T values.

When selecting a force-torque sensor controller, it is important to consider output data format, output resolution, and available software interfaces for further integration of sensorial information into the global software architecture. Common output formats include RS-232, analogue voltage, and computer bus (ISA, PCI, PC/104, VMEbus, PCMCIA, USB, and IEEE-1394). Resolution, noise rejection, and interface software vary by sensor model and manufacturer and should be checked carefully.

There are two basic force-torque sensor controller types: stand-alone and computer bus. Standalone sensor controllers are self-powered and self-contained. They typically communicate with the robot controller serial bus or by analogue voltage. Computer bus sensor controllers target specific computer backplane architecture and can be plugged into the robot's main computer. Communications are provided through software drivers or directly to I/O mapped registers or system interrupts. The computer bus force-torque sensor controller can be located inside the robot system and therefore has a much cleaner appearance (in terms of space distribution) than the standalone type.

6.3.7.3 Gyro sensor

A gyroscope sensor is a device used for measuring orientation of the upper body of a humanoid robot. It is based on the principles of angular momentum [Srinivasan 2007]. Generally, a gyro sensor is a device that provides an output in response to a rotation. The gyro

sensor is a rate sensor that measures the speed of rotation. It means that the speed of rotation is measured in “Inertial Space”. That is, it is the absolute speed with respect to the rest of the universe that is measured.

Modern gyro sensors are made using Micro-Electro-Mechanical Systems (MEMS) which integrate mechanical elements and electronics on a common silicon substrate through micro fabrication technology.

The Scale Factor of a rate sensor determines how much electrical output is given for each unit of rotational speed. Typically, this is given in millivolts per degree per second. All rate sensors are active devices – that is, they require an electrical power supply. They are not generators, (although in some instances, a generator could be used to measure rotational speed). Therefore, it is impossible to get more than the supply voltage as an output.

The frequency response of a rate sensor is like any other electronic or mechanical system – at very high frequencies the sensitivity decreases. The bandwidth is a measure of this frequency response. All rate sensors produce some noise; some are noisier than others, and noise need not always be a bad thing – indeed some noise can be beneficial. A very high bandwidth sensor will produce more noise than a sensor with lower bandwidth. Consequently, it is good practice to choose a sensor having enough bandwidth, (so as not to lose information), but not so much that the noise becomes a problem. Nevertheless, as has already been shown in chapter 5, rate gyros suffer greatly from temperature dependence bias. As we have to integrate the signal, small errors in the measurement of speed of rotation are integrated into progressively larger errors in position, known as drift phenomenon. Therefore, an additional data processing system to remove the drift related error related is needed.

6.3.7.4 Accelerometer sensor

An accelerometer is a device for measuring acceleration and gravity induced reaction forces. Single- and multi-axis accelerometer models are available to measure acceleration as a vector quantity or just one or more of its components. Accelerometers can be used to sense inclination, rotation, vibration, and shock. Nowadays accelerometers are increasingly present in portable electronic devices and in the automotive industry. In humanoid robotics are used for measuring the level of torso rotation.

Modern accelerometers are one of the simplest micro electro-mechanical system (MEMS) devices possible. It consists of little more than a suspended cantilever beam or proof mass (also known as seismic mass) with some type of deflection sensing and electronics. As reaction force on the accelerometer causes it to accelerate, the beam or the proof weights deflect. In some models the deflection is measured as an analogue signal. Some instead, take a digital approach and employ circuitry. Once the beam or mass has deflected sufficiently to reach a deflection point, an electrical pulse is generated to restore the beam or mass to the neutral mass. This pulse is also sent the outside world to provide a delta speed change over the update time periods.

Other methods of building MEMS based accelerometers are also available. In some methods a small heater at the bottom of a very small dome heats the air inside the dome to cause it to rise. A thermocouple determines where on the dome the heated air reaches and the deflection off the center is a measure of the acceleration or force applied to the object. Nowadays, MEMS accelerometers are available in a wide variety of ranges up to thousands of g’s.

Finally, it worth mentioning, that when accelerometers are used to measure an angular position in the inertial system (the case of a humanoid robot) an estimation error appears. Therefore, an additional source of inertial information and complex estimation algorithms should be implemented (chapter 5).

6.3.8 Power control

6.3.8.1 Requirements

Power control deals with routing electric power, controlling its quality, and controlling the devices attached to a power line of a humanoid robot. Also this system is associated with the efficient conversion and conditioning of electric power by static means from its available input form into the desired electrical output form [Batarseh 2003] in the control system. The basic element of a power control is a power supply.

In a humanoid robot, the power control system is the backbone that provides the correct functioning of the entire motion control system. Power requirements of a motion control system are very variable. This means that in different stages of robot motion power consumption varies. The motion control system may require a abrupt instantaneous change in power. Therefore, the power control system for humanoid robots should provide the following functionality:

- Autonomy. As a robot operates in autonomous mode, it needs power autonomy as well.
- Adequate voltage and power output from the power supply
- Provide required power peaks at every level of power supply charging
- Electrical circuits of the system should support the required current
- Small weight and size of components in order to mount these in the humanoid's robot body

In order to define the power control system parameter the basic electrical elements of the entire robot control system, and its electrical requirements such as required voltage, current and power (steady and peak) must be determined. Once these have been established it is possible to start selecting the power supply and designing power conditioning subsystems.

6.3.8.2 Power supply

Power supply means the source of electrical power. Since a humanoid robot is an autonomous system it needs an autonomous power supply. A rechargeable battery is the best solution. In electronics, a battery is two or more electrochemical cells connected in series which store chemical energy and make it available as electrical energy. There are many different types of electrochemical cells, including galvanic cells, electrolytic cells, fuel cells, flow cells and voltaic piles. A battery's characteristics may vary due to many factors such as its internal chemical structure, current drain and temperature. It is possible to recharge a special type of batteries. Rechargeable batteries can have their chemical reactions reversed by supplying electrical energy to the cell, restoring their original composition [Fink 78].

There are a few main types of rechargeable batteries that can be implemented within the power control system of a humanoid robot:

- Nickel-cadmium battery (NiCd): Ideal for motorized equipment (motion control applications) and other high-discharge, short-term devices. NiCd batteries can withstand even more drain than NiMH. However, their electrical current rate is not high enough to keep a device running for very long time, and the memory drains the battery far more severely.
- Rechargeable alkaline battery: Uses chemistry similar to non-rechargeable alkaline batteries and are best suited for similar applications. Additionally, they hold their charge for years, unlike NiCd and NiMH batteries.
- Nickel-metal hydride battery (NiMH): Best used for complex high-tech devices. NiMH batteries can last up to four times longer than alkaline batteries because NiMH can withstand high current for a long while.
- Lithium ion battery: commonly used in digital devices such as mobile phones and cameras. These devices have a very long life (up to ten years in wristwatches). Main disadvantage is that these batteries are very expensive, especially when high currents are required by the application.

The cells in a battery can be connected in parallel, series or both in order to increase the total output power. A parallel connection of battery cells has the same voltage as a single cell, but can supply a higher current (sum of currents of each cell). A series connection provides the same current as a single cell but its voltage is the sum of the voltages of all the cells.

6.3.8.3 Power conditioning

A power conditioner is a subsystem of a power control system intended to improve the "quality" of the power that is delivered to electrical load equipment. This is a device that acts in one or more ways to deliver voltage of the proper level and characteristics to enable load equipment to function properly. In some usages, "power conditioner" refers to a voltage regulator with at least one other function to improve power quality (e.g. noise suppression, transient impulse protection, etc.) [Dugan 2003].

Power conditioners are usually determined by the application; therefore they can have specific functionality and size. Some power conditioners provide only minimal voltage regulation while others provide protection from many power quality problems. In general, power conditioning subsystem should provide:

- control of the power supply voltage
- conversion of the power supply output to the appropriate type and magnitude
- delivery of a high power factor (grid applications)
- little to no harmonics
- efficiency under all possible operating conditions

6.3.9 Proposed hardware control architecture scheme

Summarizing everything stated above leads to a proposal for a general design of the hardware architecture for humanoid robots.

The proposed hierarchical architecture has a large level of scalability and modularity by dividing the control task into *Organizational, Control, Device and Sensory levels* (Figure 6.8).

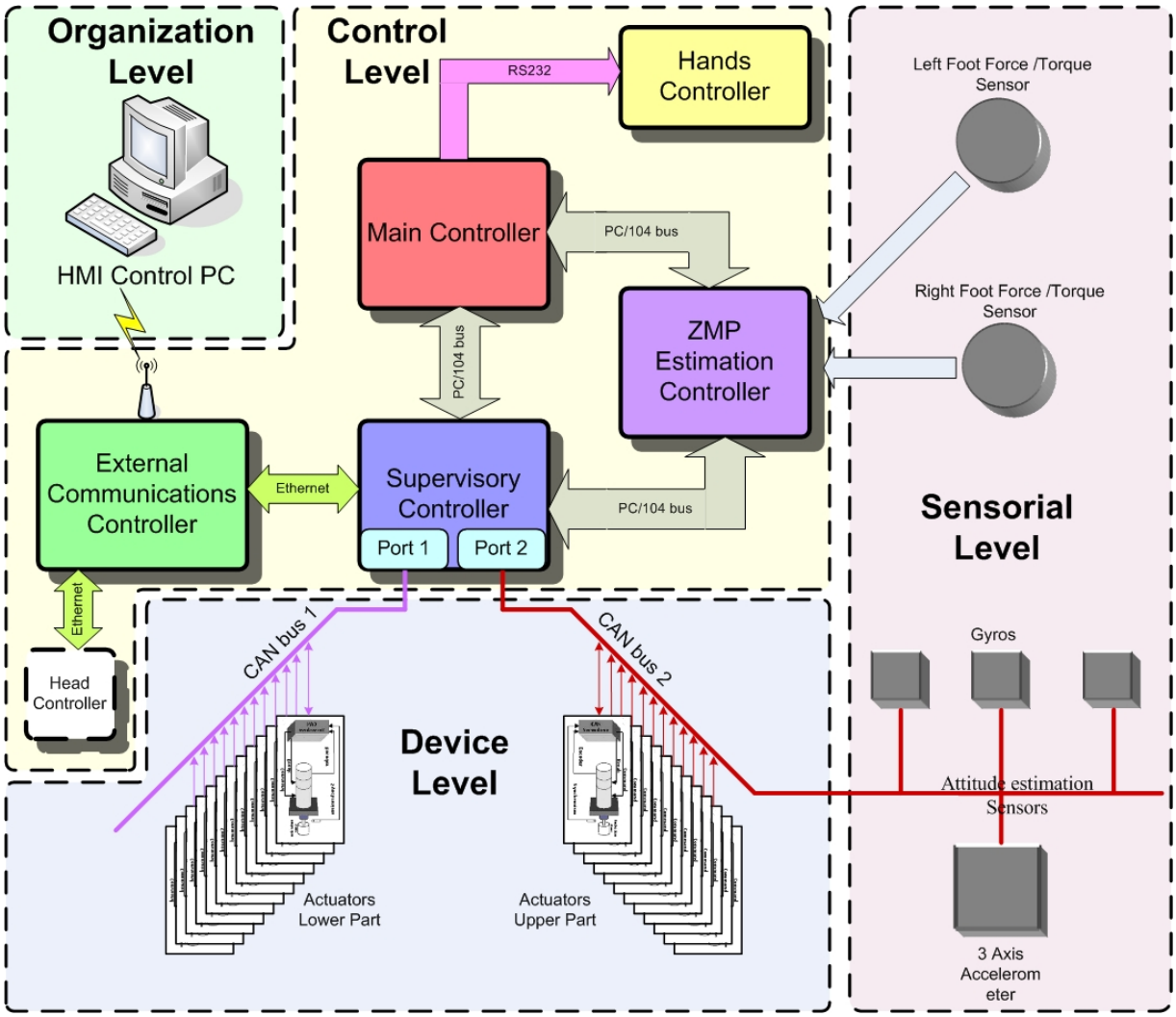


Fig. 6.8: Hierarchical Hardware architecture

In the *Control level*, each layer is represented as a controller centered on its own task, such as external communications, motion controller's network supervision or general control.

In the *Device level*, each servo drive not only closes the servo loop but calculates and performs online trajectory, synchronizes with other devices and can execute different movement programs located in its memory. These devices are located near the motors where they benefit from less wiring which is one of the requirements for energy efficiency, light weight and ease of cabling.

Advanced and commercially available motion controllers can be used in order to reduce development time and cost. Continuous evolution and improvements in electronics and computing have already made it possible to reduce the controller's size for use in the humanoid development project. Furthermore, they offer the advantage of applying well supported and widely used devices from the industrial automation field, and bring the widely used and well supported standards into the humanoid robot development area.

A stack of PC/104 modules can be considered as different controllers executing its tasks and communicating via PC/104 bus (Appendix A), thus, organizing the *Control level* of the motion control architecture. The Main controller is a commercial PC/104+ single board computer because of its small size and low energy consumption. It was selected instead of a DSP controller because it has different peripheral interfaces such as Ethernet and RS-232, and an easy programming environment. There are a great variety of additional extension modules for PC/104+ bus such as CAN-bus, digital and analogue input-output and PCMCIA cards. Selection criteria were fast CPU speed, low consumption and availability of expansion interfaces. The Main Controller provides general synchronization, updates sensory data, calculates the trajectory and sends it to the servo controllers of each joint. It also supervises data transmission for extension boards such as Supervisory Controller and ZMP (Zero Moment Point) Estimation Controller via PC/104+ bus.

The Communication Supervisory Controller uses a network bus to reliably connect distributed intelligent motion controllers with the Main Controller.

The motion control domain in the industrial automation field is rather broad. As a consequence of the breadth of the domain, communications standards to integrate motion control systems have proliferated. The most appropriate solution for the humanoid robot motion control system design seems to be the use of the CAN based standards. The CAN bus communication for the *Sensory leve*l and the CANOpen protocol on top of CAN bus for the *Device level* of communication.

Therefore, the control system adopted for a humanoid robot design is a distributed architecture based on CAN bus. The CAN bus also was chosen because of its characteristics: bandwidth up to 1 MBit/s which is enough speed to control the axes of a humanoid robot, large number of supported nodes (humanoid robots may have up to 40-50 controllable D.O.F.), differential data transmission that is important for the reduction of EMI effects caused by electric motors and the for the capacity for other devices such as sensors to reside on the same control network.

In the *Device level*, the controller network is divided into 2 independent CAN buses in order to reduce the load of the bus. The Lower part bus controls the both legs nodes and the Upper part bus controls the two arms and the trunk nodes. To unify the data exchange inside the robot, the attitude estimation sensory system is also connected to the Upper part CAN bus. By this, the communication speed of the CAN bus is 1MBit/s. The synchronization of both parts is carried out by the Supervisory Controller in the *Control level* of automation.

The External Communications module provides the Ethernet communication on the upper (Control and Organization) levels of the automation with the head electronics which comprises the independent vision and sound processing system. It also provides wireless communication with the remote HMI control PC which sends upper level operating commands to the humanoid robot. It can be noted that proposed architecture fully complies with the industrial automation standards on the design of the motion control system.

6.4 Software architecture design

6.4.1 Considerations on software architecture design for humanoid robots

The software architecture of a robot can be defined as "the structure or structures of the system, which comprise software components, the externally visible properties of those components, and the relationships between them" [Bass 2003].

While there are numerous definitions of software architecture, at the core of all of them is the notion that the software architecture of a system describes its internal structure. This structure illuminates the top level design decisions, including such things as how the system's interacting parts are composed, indicating where the main pathways of interaction are, and what the key properties of the parts are. Additionally, an architectural description includes sufficient information to allow high-level analysis and critical appraisal [Garlan 2000].

Software architecture typically plays a key role as a bridge between posed requirements and future implementation of a system as a program. By providing an abstract description of a system's elements, the architecture highlights some important properties, while hiding others less important. In the ideal case this representation provides a complete guide to the implementation of the overall system.

Basic requirements of the software architecture for humanoid robots are:

- The architecture should have an adequate structure for providing the functionality of an entire system. It is necessary to take into account the dynamics and general behavior of the entire robot while the system is still under development
- The architecture should fulfill functional conditions and provide full functionality for motion control tasks. These include efficiency, reliability and security associated with actual required functionality.
- The architecture should be open, flexible and easily expandable for future modifications

The creation of an architectural base for developing the software for robotic systems is essential to the success of any further implementation. At this level of design, all details (code writing) are excluded, only the basics of the software architecture for humanoid robots will be discussed. Following sections will present proposed software architecture and the discussion of its principal components.

6.4.2 Proposed software scheme

At the architectural level a humanoid robot can be compared to a plant were the shop floor consists of a series of cells (intelligent motion controllers and a sensors) managed by controllers (Main Controller, Communication Supervisory Controller, etc.). In general, there are two principal control tasks for the motion control system of a humanoid robot. The first goal is to control all automation and supervise the data transmission. And the second goal is to control and monitor the entire robot's state to detect failures as soon as possible and to report on performance indicators. In the automation industry, in order to perform these two different control tasks, a

PLC with its own internal program and a SCADA system are used. In this context, a humanoid robot can be provided with a software system that allows the implementation of the industry control concepts but at another, more sophisticated level. The proposed software architecture is based on the Server-Client model (Figure 6.9).

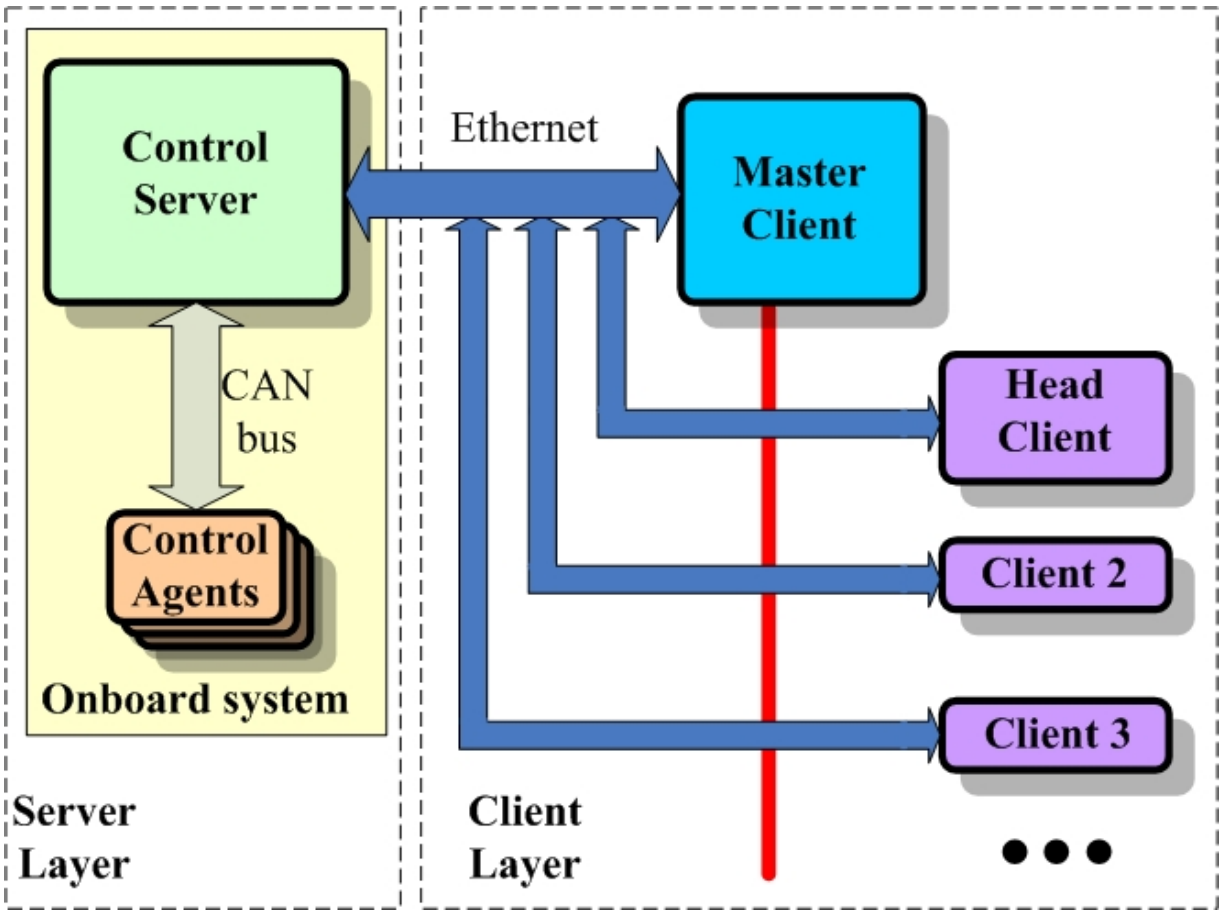


Fig. 6.9: Software architecture for humanoid robot motion control

Proposed architecture consists of two layers: the Server Layer and Client Layer. According to the Server-Client model, the humanoid robot is controlled by the passive Server, which waits for requests and upon their receipt, processes them and then provides replies for the Client. But on the other hand, the Server controls all Control Agents which are responsible for humanoid robot motion control. Control Agents reside on the CAN bus network. In that case, the Control Server is a network master for Control Agents that perform their operations (motion control or sensing) and reply to the Server.

Master Client is a principal application residing in the external computer and providing HMI for the user. For security, the Control Server accepts the connections of other clients, such as the Head Client, responsible for the human-robot interaction, only if the Master Client allows it. If the connection is accepted, the Master Client only supervises the humanoid robot state and data transmission between the robot and other Client, but always in the case of any conflict it has the main priority. The system may have more than one client connected to the robot simultaneously.

The interactions between the elements of the architecture are shown in figure 6.10. The connection between the Server and Client can be made using sockets.

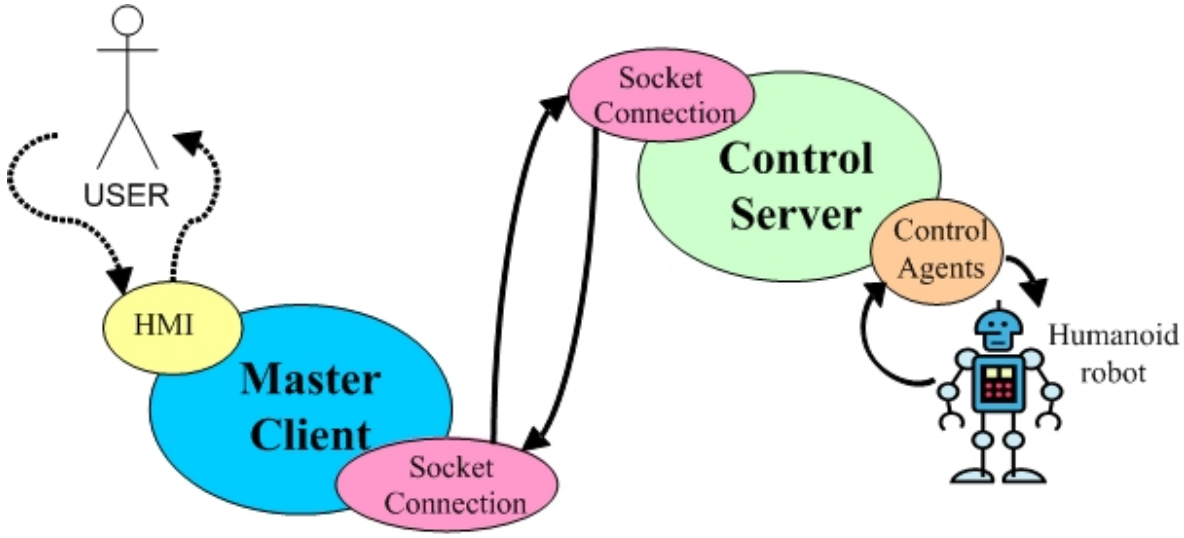


Fig. 6.10: Software architecture interaction

The interaction is actually quite simple. The user feeds commands into a workstation running the Master Client application, through a graphical interface. Then the workstation feeds commands through a socket to an onboard control system. The onboard system then controls the robot via execution of Control Agents and feeds information back to the workstation computer that then displays the necessary information back to the user through the user interface.

6.4.3 Control Server

6.4.3.1 Requirements

The server is an application that performs services for connected clients as part of a client-server architecture. A server application is an application program that accepts connections in order to service requests by sending back responses.

The onboard Control Server will un-marshal the data received from the client program through the socket connection. The system will then call the corresponding methods for activating a Control Agents. The corresponding methods will communicate with the Agent and send commands. The onboard Control Server system has several threads; network thread, command thread, and data thread. The network thread deals with the communication between the onboard system and the workstation system with running Master Client application. Next the command thread handles the command queue and passes all the commands to the bottom control level (it may be the Control Agent directly or a control thread of the server) which controls the motion of a humanoid robot. The data thread then manages, parses and gets all the requested data from the robot and passes it to the network thread. And finally, the control thread provides all upper level control tasks: acquires sensorial data, computes stabilization control, modifies motion patterns

and synchronizes all Control Agents. The interaction between the basic threads of the Control Server is presented in figure 6.11.

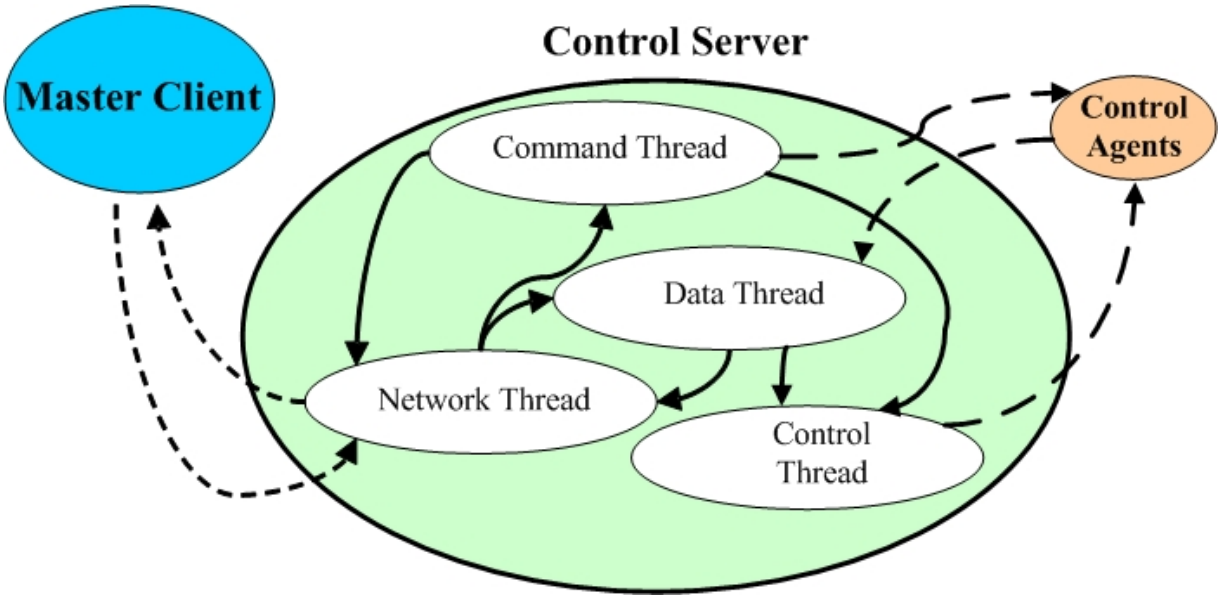


Fig. 6.11: Control Server threads

Different processes running in the humanoid robot Control Server require different processor times. For example communication with an external client has less priority than the stabilizing control of a humanoid. Moreover, it executes all control tasks consecutively based on their priority. To sum up all of the above, the basic requirements for the Control Server implemented in the motion control architecture of a humanoid robot are:

- Provide command thread to the Control Agents
- Provide data thread to a client application
- Provide the network thread for a data exchange between different applications
- Provide control thread
- Provide the task priority
- Provide the synchronization and consecutive execution of different control process within required sample time

All threads inside the Control Server (figure 6.11) are implemented as software modules. Command and data threads are the simple functions that provide the data flow inside the Control Server. The Network module will be presented after the communication infrastructure design. The principal thread here is the Control thread or Control Area of the server which is considered in detail below.

6.4.3.2 Sample Time

When a humanoid robot is working in the operational interaction mode (walking, manipulating objects, etc.), there are several computing and communication tasks that the Control Server must perform cyclically and rapidly enough to avoid any possible loss of the control. The periodic (with period Ts) chain (Figure 6.12) begins with the sensing task, taking the time interval t_{att} for attitude estimation gyros and accelerometers readings and t_{zmp} for ZMP force-torque sensors readings.

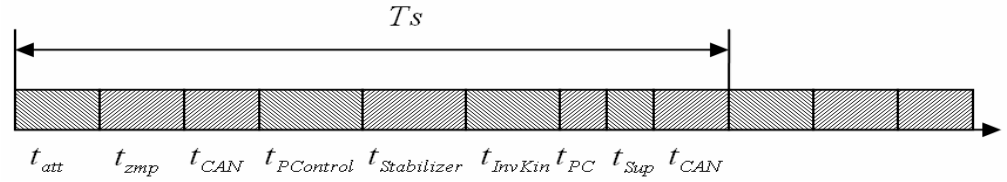


Fig. 6.12: Main computing and communication tasks of a humanoid robot

These tasks are followed by the tasks performing CAN bus communications, Posture Control and then Stabilizer and inverse kinematics computing, internal PC bus communications, Supervisory controller, and CAN bus transmission of new reference for each joint of a humanoid robot.

The period Ts should be small and compatible with the dynamics of humanoid robot movement. On the one hand, Ts cannot be made arbitrarily small because various computing and communication tasks with execution times t_{att}, t_{zmp}, t_{CAN}, t_{PC}, $t_{Stabilizer}$, etc., cannot be made arbitrarily small. Also, a small value for Ts would generate too many messages in communication lines (RS-232, PC bus, CAN bus) that could overload it. On the other hand, Ts cannot be made arbitrarily large because of the dynamics of the robot (Nyquist criterion).

Thus,

$$Ts > t_{att} + t_{zmp} + t_{CAN} + t_{PControl} + t_{Stabilizer} + \\ + t_{InvKin} + t_{PC} + t_{Sup} + t_{CAN} \tag{6.1}$$

And

$$Ts < \frac{1}{2 \cdot Fr} \tag{6.2}$$

Where Fr is the highest movement frequency on any robot link.

Assuming that the robot walks at about the same speed as a normally walking human, $Fr = 2$ Hz.

Thus,

$$Ts < 250 \text{ ms} \tag{6.3}$$

A reasonable value for the sum of the time intervals taken for all computations and calculations is $0.1 \cdot Ts$, thus

$$t_{att} + t_{zmp} + t_{CAN} + t_{PControl} + t_{Stabilizer} + \\ + t_{InvKin} + t_{PC} + t_{Sup} + t_{CAN} = 25\ ms \tag{6.4}$$

This upper limit of the sample time is a strict time requirement given the complexity of the computing and communication tasks to be performed within this time limit. Therefore, the value of *Ts* should be taken closest to this value.

6.4.3.3 State diagram

As a PLC in the automation industry, the Control Server is designed and programmed as finite state automata with a sample time *Ts* . Figure 6.13 shows the state diagram and Table 6.2 shows the state transition events of the humanoid robot Control Server functioning.

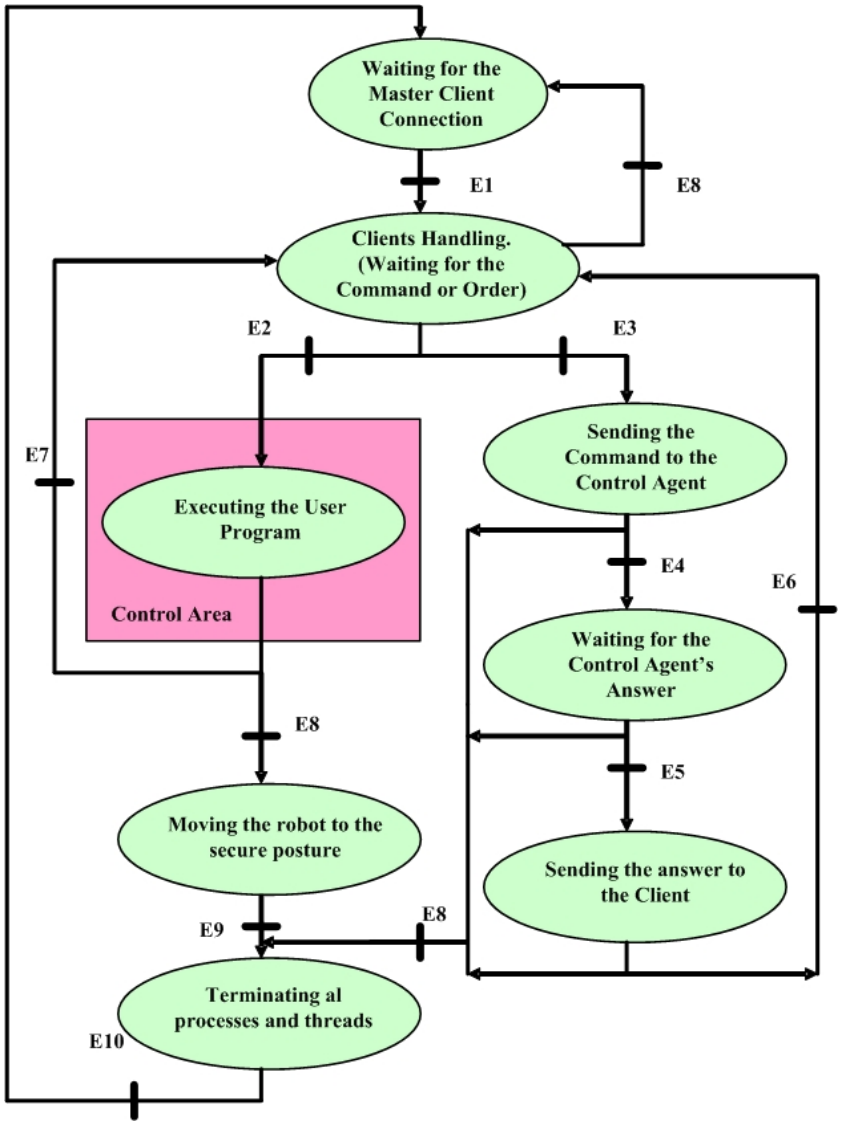


Fig. 6.13: Server functioning state diagram

Event	Event Description
E1	The Client is connected
E2	An Order has arrived
E3	A Command has arrived
E4	A Command is sent to the Control Agent
E5	Agent's reply has arrived
E6	An Answer is sent to the Client
E7	The User Program is successfully terminated or an Error Event has occurred
E8	Connection with the Client is lost
E9	The Robot is staying in the secure position
E10	All processes are terminated

Table 6.2: Server Events

Two basic types of the incoming data are processed. A command is the simple data, which can be executed by one Control Agent. An order is a complex command which needs the simultaneous action of the many Control Agents and sensors that the humanoid robot possesses. After the connection of the Master Client, the humanoid robot stays in the Client Handling state waiting for an order or a simple command. The arrival of an order launches the Control Area of a robot where the control thread guided by a User Application is usually executed. The Control State is a core of the humanoid robot Control Server software. It performs data transmission between each Control Agent, the sensory system and the Control Server. It performs the trajectory execution in synchronized multi-axis walking applications, controls the posture and ZMP errors in the dynamic walking mode, reads the sensors state etc. In order to create a user program, the control architecture provides a set of standard methods.

6.4.3.4 Control Area

The Control area consists of different modules which execute motion control for humanoid robot stable biped locomotion. All tasks can be grouped by their time requirements. The joint control and trajectory execution is performed by Control Agents, as these operations are subject to operational deadlines from event to system response and therefore should be implemented in hard real-time. Here, sensors and control feedback should be less than 0.4 [ms] (chapter 4). In order to ensure the high performance of these operations, Control Agents were taken out of the Control Server tasks. This means that the Control Server works in soft real-time conditions. A soft real-time system will tolerate some lateness in operations but fast response and high performance is preferred.

The main soft real-time task of a server is the execution of the Locomotion module which controls stability and generates a target position set table which is sent to the Control Agent for execution by the motion controller of each joint of the robot. The CAN bus module provides the communication between the Locomotion module and Control Agent. The sample time for sensors and control feedback in this case should not exceed 25 [ms]. (section 6.4.3.2)

In addition to locomotion control tasks, there are other tasks executed in the control area of a server. They are related to the generation of reference motion patterns and communication with

an external client application. These modules are not critical to the robot and do not require real-time operations because there is no deadline for their execution. Figure 6.14 shows the principal modules of the control area.

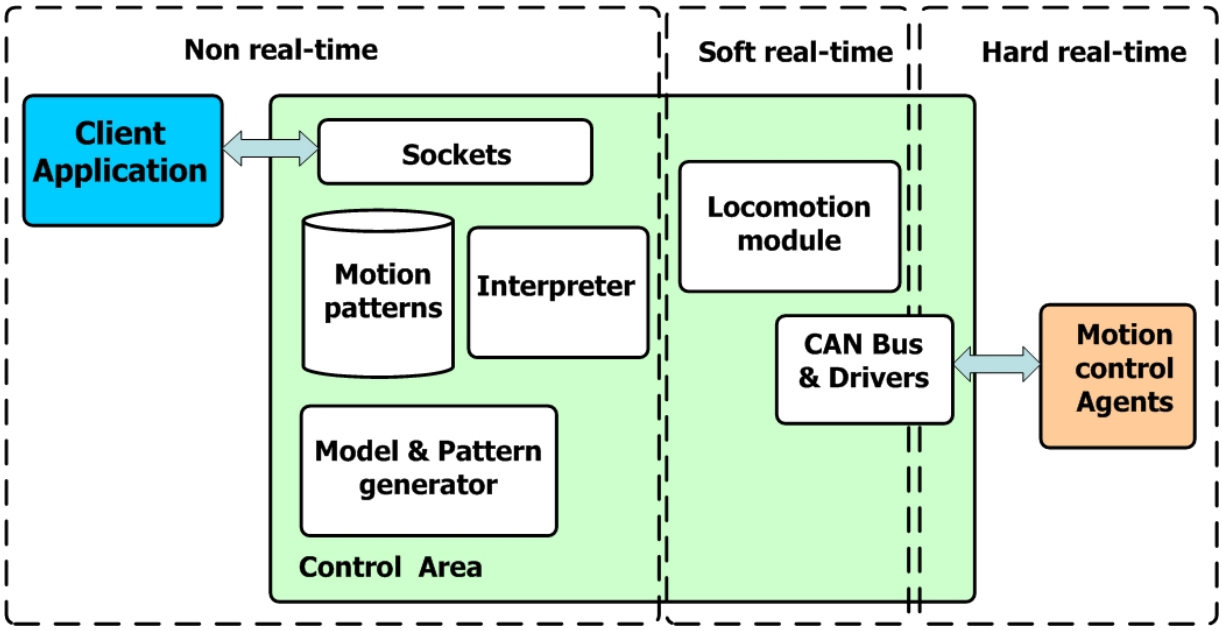


Fig. 6.14: Control area modules

All modules are constructed as an open library of functionalities for humanoid robot motion control. A user can manage these libraries to add functionality to the system. For example, to control special types of motion (running locomotion, squatting locomotion, etc.) or to make changes in the already existing control procedures.

6.4.3.5 Application Programming Interface

An application programming interface (API) is a source code interface that the developer provides to support requests made by software architecture. Proposed software architecture provides the API library of the C-based function to work with the robot and generate user motions and control procedures, not only for walking, but for implementing different human-robot cooperation tasks. It can be said that these functions provide the link between the robot's hardware and the motion control software (Figure 6.15).

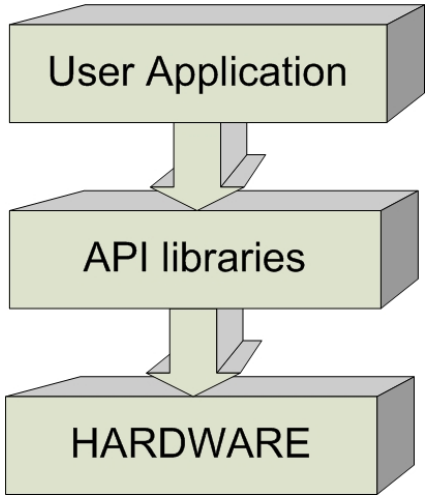


Fig. 6.15: The link between user application and a humanoid robot

The software or user program that provides the functionality described by an API is an implementation of the API. The API is a set of abstractions (methods) that specify an interface and the behaviours of the functions specified in that interface. A user program specifies how these behaviours may be implemented. The code below shows the simple user program implemented with proposed software system:

```
// Motion program example
#include "control.h"
#include "sensors.h"
#include "dynamic_gate.h"

main (void)
{
        EnableRobot();
        ReadJointsPosition();
        Sincronize();
        if ( Start() )
                do
                {
                        PostureControl();
                        DynamicGate(Gate1);
                        if (ChangeGate)

        DynamicGate(Gate2);
                }
                while (!Stop())
        DisableRobot();
}
```

Fig. 6.16: Motion program example

The example shows how simple humanoid robot motion can be programmed using developed API functions. At the beginning, the synchronization procedure for every joint is executed, and then the motion is started. The robot will change the gait (walking mode) on the user's request.

6.4.4 Control Agents

6.4.4.1 Requirements

The Control Agent is a piece of general software architecture that acts for a main control process in a relationship of agency [Nwana 96]. Such functionality implies the authority to decide which (and if) action is appropriate. This makes the Control Agents independent actors of the motion control architecture. These agents are specialized motion control programs running on the physically distinct machines. Based on the distributed hardware architecture it is clear that these Control Agents are software programs implemented for intelligent motion controllers which control every joint of a humanoid robot.

The main task of the Control Agent is to provide the execution of a motion of a single joint of a humanoid robot. The distributed Agent's software should be able to receive and send the data from and to the CAN bus connection. The Agent will then call the corresponding methods for motion interpolation and control. Thus Agent software systems are several threads: communication thread, motion interpolation thread, sensorial data thread and motion control thread (figure 6.17).

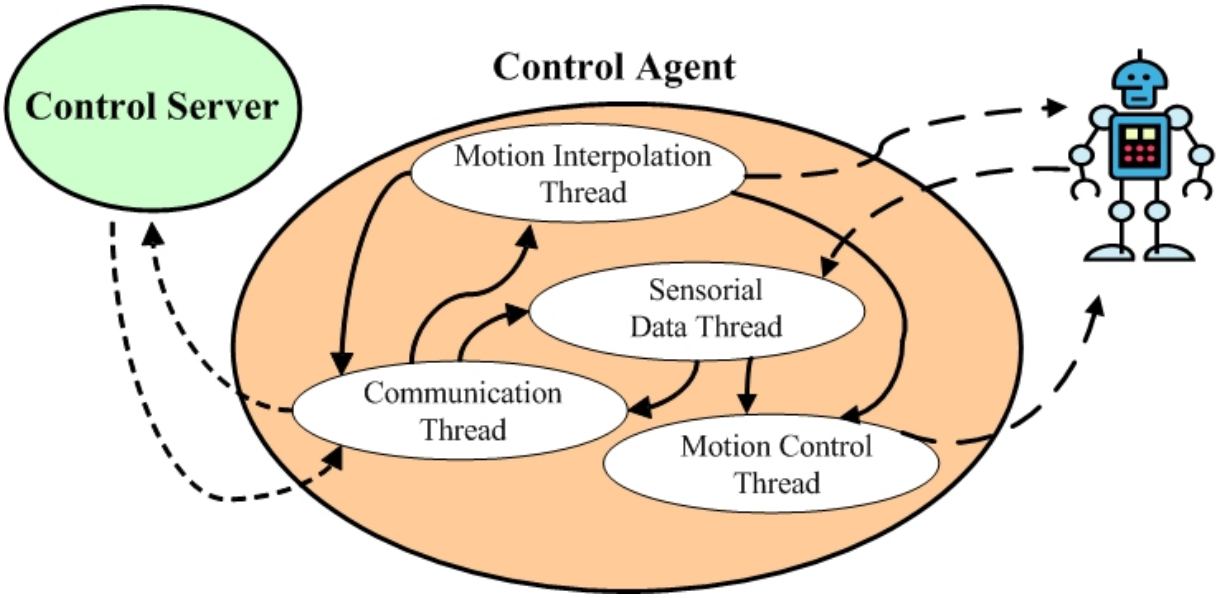


Fig. 6.17: Control Agent threads

As can be observed from figure 6.17, the Control Agent in this scheme performs low level server functions and the Control Server performs client functions. The communication thread deals with the communication between the Agent software and the onboard main computer by running the Control Server application. The interpolation thread handles the internal target positions table and creates the continuous trajectory of a joint. The data acquisition thread provides the queue of actual motion and sensorial data and passes it to the control thread (it may be the Agent control thread directly or a control thread of the server) which controls joint's motion of a humanoid robot.

The Control Agent's software operates at the bottom level of the software architecture interacting closely with the hardware. Therefore, the total correctness of an operation depends not only upon its logical correctness, but also upon the time in which it is performed. The completion of an operation after its deadline is considered useless - ultimately, this may lead to a critical failure of the complete system. This leads to the need for hard real time requirements for a Control Agent's software implementation. Summarizing everything stated above, the requirements for the Agent's software are as follows. It should:

- Provide hard real-time task execution
- Provide data (sensorial, motion) acquisition
- Provide interpolation of a trajectory
- Close joint motion control loop
- Provide communications with the Control Server

Since Control Agents are well suited to carry out their required function for their clearly specified task, they are designed to be decoupled and it becomes easy to have them executed as independent threads and on distributed processors of intelligent motion controllers. Thus they become distributed agents and the considerations of distributed computing apply. Agent code is very easy to implement in a distributed fashion and the final solution scales well.

All threads inside the Control Agent (figure 6.17) are implemented as software modules running on the intelligent motion controller hardware. The sensorial data thread module is a simple function that acquires the data from the hardware. The communication module will be presented later, in the communicational infrastructure design section. The joint control module was presented in details in chapter 3. The principal module, still not covered in design process is the motion interpolation thread which will be considered in detail below.

6.4.4.2 Motion interpolation

From the software and control implementation points of view, biped motion of a humanoid robot can be considered as a set of position references for its joints. Control Agents managing joints motion, implement a third-order interpolation between the target position data points (motion pattern) provided by the Control Server. Let T_a be the sample time a Control Agent functions (corresponds to the hardware sample time of a motion controller). Therefore, the sampling time of the trajectory (motion pattern) will be:

$$T = m \cdot T_a \tag{6.6}$$

where m is the integer parameter that relates T_a and T. This parameter regulates the velocity of motion pattern execution and, therefore, the walking speed of a humanoid robot. For $m = 1$, no interpolation is required. For $m > 1$, there are certain sampling instances of the position controller for which the path command must be interpolated, using a third order polynomial interpolation.

The Control Server provides position points $P(k),\ k=1\ldots N$ of the previously computed motion pattern to the Agent.

The Agent calculates the speeds $V(k),\ k=1\ldots N$ for points $P(k),\ k=1\ldots N$ as follows:

If k is an ordinary point in the path:

$$V(k)=\frac{P(k+1)-P(k-1)}{2\cdot T} \tag{6.6}$$

If k is the first programmed point in the path:

$$V(k)=\frac{P(k+1)-P(k)}{T} \tag{6.7}$$

If k is the last programmed point in the path:

$$V(k)=\frac{P(k)-P(k-1)}{T} \tag{6.8}$$

For each motion segment, five requirements must be satisfied:

- Start position P_0
- End position P_T
- Start speed V_0
- End speed V_T
- Time interval between two positions (sample time) T

These five requirements exactly suffice for solving the third-order interpolating polynomial. If t_0 denotes the starting time and T denotes the length of the time interval, the position for $t\in[t_0,t_0+T]$ is given by the third-order interpolating polynomial:

$$P(t)=a(t-t_0)^3+b(t-t_0)^2+c(t-t_0)+d \tag{6.9}$$

The speed then is given by:

$$V(t)=3a(t-t_0)^2+2b(t-t_0)+c \tag{6.10}$$

The four parameters a, b, c and d are unknown and can be solved using the following four linear equations:

$$P(t_0)=P_0\Rightarrow d=P_0 \tag{6.11}$$

$$V(t_0)=V_0\Rightarrow c=V_0 \tag{6.12}$$

$$P(t_0 + T) = P_t \Rightarrow P_t = aT^3 + bT^2 + cT + d \tag{6.13}$$

$$V(t_0 + T) = V_t \Rightarrow V_t = 3aT^2 + 2bT + c \tag{6.14}$$

Solving equations (6.12)-(6.14) it is possible to obtain a and b coefficients:

$$a = \frac{(V_0 + V_t)T + 2(P_0 - P_t)}{T^3} \tag{6.15}$$

$$b = \frac{3(P_t - P_0) - (2V_0 + V_t)T}{T^2} \tag{6.16}$$

By these means, the third order interpolation allows a continuous trajectory of every joint of a humanoid robot to be generated from the discrete target positions points proposed by the motion pattern. Figure 6.18 shows an example of the motion trajectory interpolation.

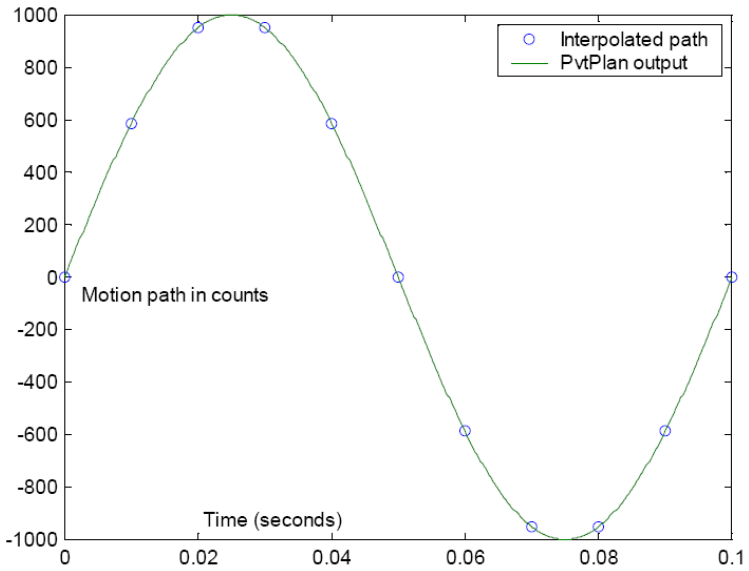


Fig. 6.18: Motion interpolation

The path interpolated by the Control Agent is shown as a solid line.

6.4.5 Server – Agent dynamical motion interaction

As mentioned above, one of the principal control tasks of a Control Agent is to provide the execution of a motion of a managed joint. The motion is provided by a set of target positions for the motor. These target positions are sent by a Control Server to the Control Agents residing in the intelligent motion controller of every joint of a humanoid robot.

The general list of target positions is stored in the memory of the Control Server. It is based on the previously computed motion pattern and is dynamically changed in accordance with the walking stability needs of the robot. The target positions are the angular positions for each joint of a robot taken with a desired sample time (Figure 6.19).

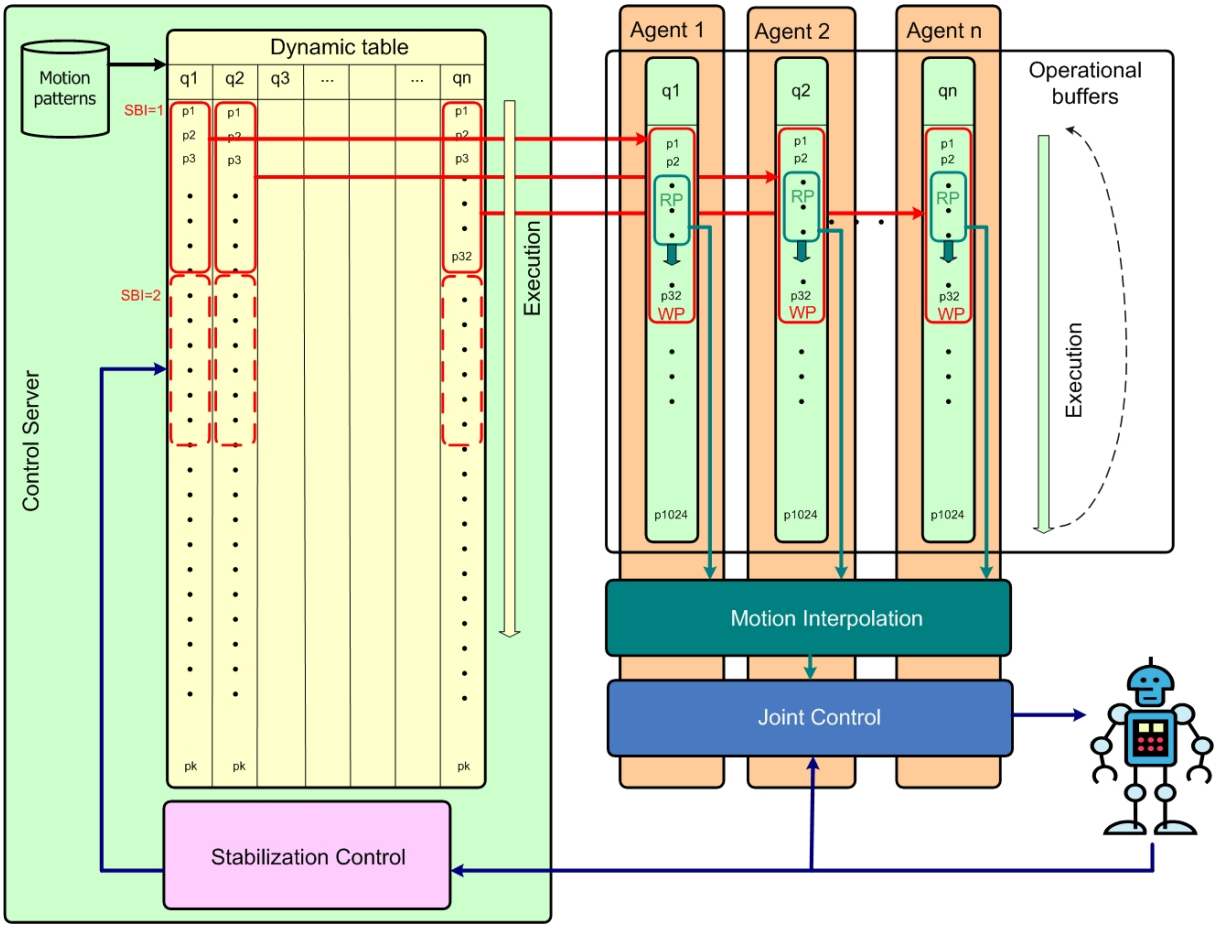


Fig. 6.19: Server –Agents dynamic interaction

In practice, the dynamic table of a motion can have many data points. This means that it is possible to pre-compute a multitude of consecutive motions which a humanoid robot should perform. As the table in this case will have a large quantity of data, it can't be stored directly in the motion controller of each joint.

One of the possible ways to organize the operation in this case, is to provide a real-time transfer of every point of the trajectory from the Control Server directly to the Agent. The great disadvantages of this method are: the need for a very high bandwidth for the communication bus and a real hard real-time operation of the Control Server.

The solution is to divide the entire trajectory into small parts (sub buffers) containing, for example, 32 target position points and organize the operational buffers inside the Agent's program. The operational buffer of each Agent's program is a memory zone which can contain up to 1024 target position points. In the dynamic operational mode, the Server will send a small sub buffer to each Agent and place it in the operational buffer for continuous execution. As the size of the operational buffer is a 1024 target position points, it will be executed in a cyclic mode.

The Agent computes a motion segment interpolation taking one current target point and the values of the previous one along with the next target position as was described in the previous section. After the interpolation is computed, it is sent to the robot motion controller in order to generate control torque for the motor.

As was discussed above, in order to provide stable biped locomotion it is necessary to implement joint and stabilization controls. The joint control is implemented in the Agent program and the stabilization control is implemented in the Control Server (Developed API allows to implement other control modules in the future, extending the functionality of the motion control system). Joint control closes the servo control loop. The stabilization control provides the walking stability of a humanoid. It receives the data about the current state of a robot and computes the corrections thus modifying the dynamic table. These modifications are sent to the Agent after a low buffer request is received.

The dynamic motion mode is described in the following flowcharts:

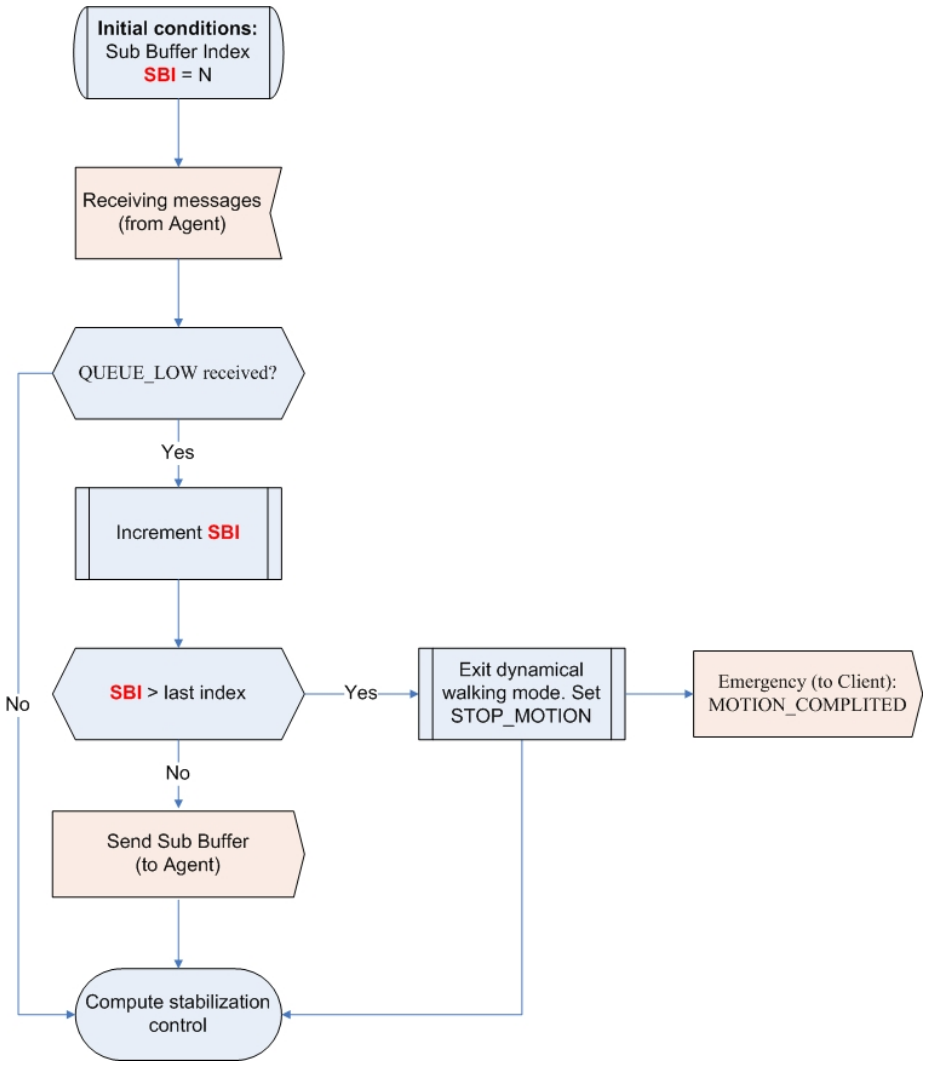


Fig. 6.20: Control Server dynamical motion mode flowchart

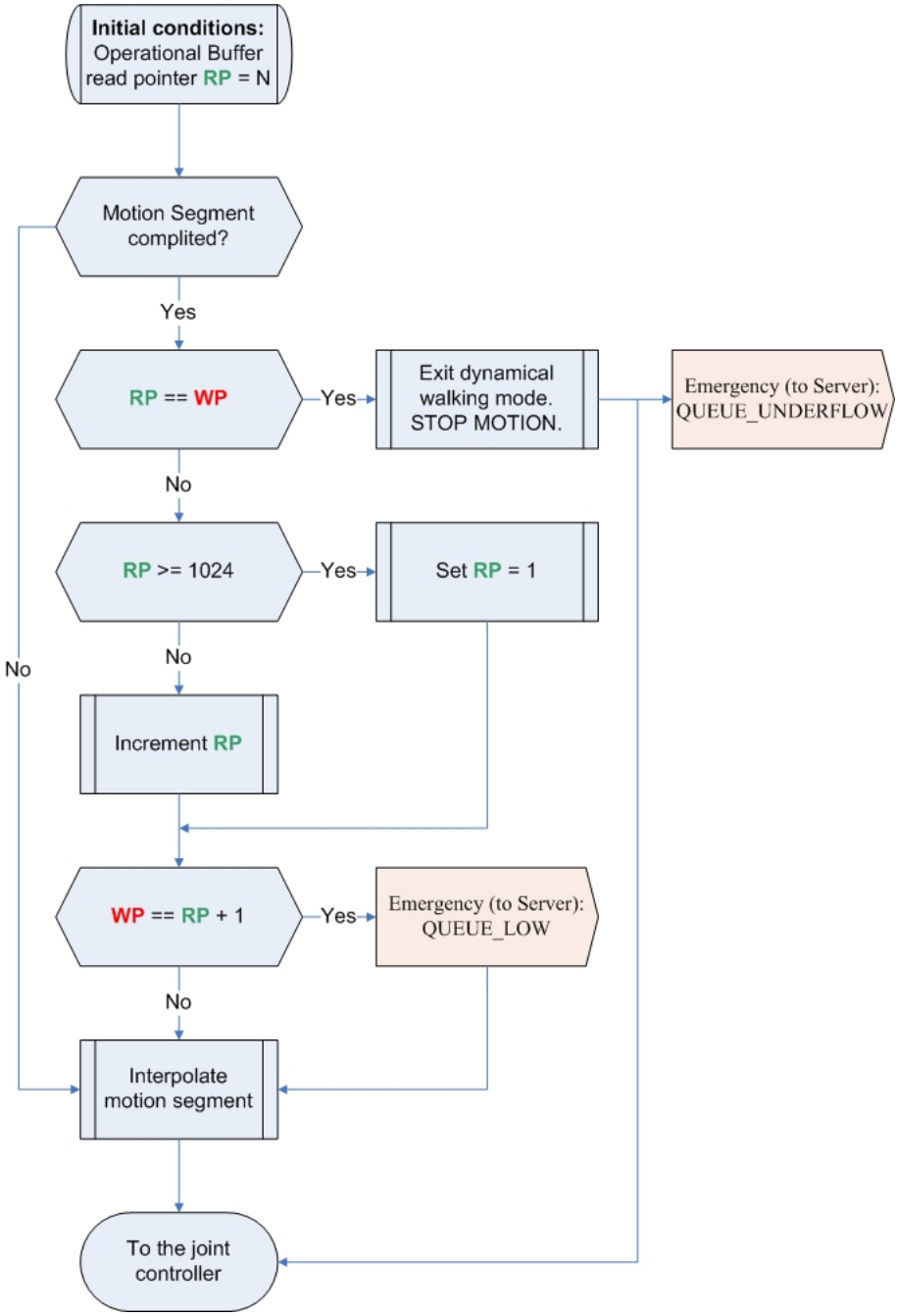


Fig. 6.21: Control Agent dynamical motion mode flowchart

The Control Server (Figure 6.20) uses a variable - the Sub Buffer Index (SBI) to save the current position of the control in the dynamic table, i.e. indicate the sub buffer under control at the moment. At each sample time, the Server receives messages from the Agents indicating their current state. If a QUEUE_LOW message is received, the Server increases the SBI index and sends new sub buffer to the Agent. If the SBI index reaches its final value, it means that the dynamical table was executed successfully and the motion can be stopped. If a QUEUE_LOW

message is not received, the Server computes the stabilization control and modifies the target points of the dynamical table which will then be sent to the Agent in the next sub buffer.

The motion Control Agent (Figure 6.22) uses two pointers in order to control the data flow in the operational buffer. Read Pointer (RP) indicates the actual target point being in use in the operational table. Write Pointer (WP) indicates the position of the last point of the last received sub buffer from the Server. The program verifies whether the motion segment (computed for the actual point) was completed. If it was not completed, the Agent continues the interpolation of the segment and joint control processes. If it is completed, the program increases the RP taking the next target point to compute the interpolation. As the operational buffer has the capacity for only 1024 points, if the RP approaches the last one, it makes the jump onto the beginning of the buffer and thus it is executed in a cyclic mode. Moreover, the Agent verifies whether the RP approaches the boundary of a current sub buffer. An emergency object QUEUE_LOW is issued for the queue low event placing an order for a new sub buffer to be sent from the Server. The read pointer reaching the write pointer identifies motion queue underflow. In this case the robot stops immediately and dangerous instability of the system can be provoked.

This mode of dynamic interaction between Control Server and Agents allows every type of biped locomotion to be implemented with online motion pattern correction. It also allows the type of a motion to be changed on the fly, only by changing the corresponding part of the dynamic table. Moreover, while a fast operational buffer is executed in hard real-time in the intelligent motion controller with the Agent's software running on it, the dynamic table is modified in order to perform stabilization control of a humanoid robot with soft real-time requirements.

6.4.6 Client

6.4.6.1 Requirements

A client in computer science is an application or system that accesses a (remote) service on another computer system (server) by way of a network [Sadoski 97]. Clients can generally be classified by their data processing and storage ability as either "fat clients", "thin clients", or "hybrid clients".

A fat client is a client that performs the data processing operations itself, and does not necessarily rely on the server. A thin client is a minimal sort of client. Thin clients use the resources of the host computer. A thin client's job is generally just to graphically display data provided by a server application, which performs any required data processing. A hybrid client is a mixture of the above two client models. Similar to a fat client, it processes locally, but relies on the server for data storage.

Relying on the proposed software scheme for humanoid robot motion control (Figure 6.9) one can observe that the architecture allows the implementation of a multitude of different remote clients working with the onboard Control Server. These clients can be used for different tasks such as sending upper order motion commands to the robot, visualization and storage of motion data, simulation, etc. Moreover, realized multi-client architecture makes it possible for various clients to work simultaneously with the humanoid robot. In general, a couple of thin clients specialized in their own very concrete task can be implemented using only the data provided by the Control Server of a humanoid.

Nevertheless, one of these clients should take the role of a master in order to manage the access of other remote clients to the server's services. In order to make a flexible and universal interface between a human operator and onboard control system of a humanoid robot, the Master Client application can be provided with more functionality than only network management tasks. It can also provide the supervisory control of a humanoid robot. Therefore, the Humanoid Robot Supervisory Control System (HRSC) including HMI, motion control data logging, visualization, simulation and trending tasks as well as the Master Client functions should be developed.

6.4.6.2 Humanoid Robot Supervisory Control System

In the proposed software architecture, the Control Server is capable of accepting many clients' connections at the same time. It is evident that the Master Client, as the basic HMI of the humanoid robot, should provide and supervise the execution of the upper level control tasks related to global motion planning, collision avoidance and human-robot interaction. In general, these tasks are common for all mobile and walking robots and the design of this kind of software systems is not considered in this work. On the other hand, there are some bottom level tasks that should be supervised, such as sensory data acquisition, joint synchronization and walking stability control. These, more oriented towards automation supervisory and control tasks are processed in the Humanoid Robot Supervisory Control (HRSC) client.

The supervisory control system is a system that sends commands to a real-time Control Server to control a process that is external to the HRSC system. This implies that the system coordinates, but does not control internal processes inside a humanoid robot in real time, as there is an integrated real-time Control Server that can respond quickly enough to compensate for process changes within the time constants of the process.

The grouping of all these tasks into a single client application requires the implementation of the HRSC as a fat client with rich functionality independent from the Control Server. The advantages of a fat HRSC client technology are:

- Fewer Control Server requirements. A fat client server does not require as high a level of performance as a thin client server (since the fat HRSC clients themselves do much of the application processing). It releases the Control Server of a humanoid from auxiliary data storage and processing. This results in increasing the Control Server performance.
- Offline working. Fat HRSC client has advantages in that a constant connection to the Control Server for some operations is not required.
- Better HMI performance. Fat clients have advantages in graphical applications that would be bandwidth intensive if fully served.
- More integrity of the application. A fat HRSC client provides a multitude of functional tasks and integrates all packages needed for the motion control of a humanoid robot.

The proposed software architecture is a multitasking fat client working in the remote terminal computer and connected with the onboard system via Ethernet.

The Control Server is responsible for motion data acquisition and handling (e.g. polling motion controllers and sensors, alarm checking, calculations, logging and archiving) of a set of parameters as well as motion control. The HRSC client requires the data or changes control set

points sending commands to the Server. The arrival of a command launches its execution procedure. This consists of interpreting a command and/or transmitting it to the Control Agent. When the answer (usually containing motion and sensorial data) from the Agent is received by the Server, it is converted and transmitted to the HRSC client to be processed and visualized using the HMI. Figure 6.22 shows the architecture of the HRSC client.

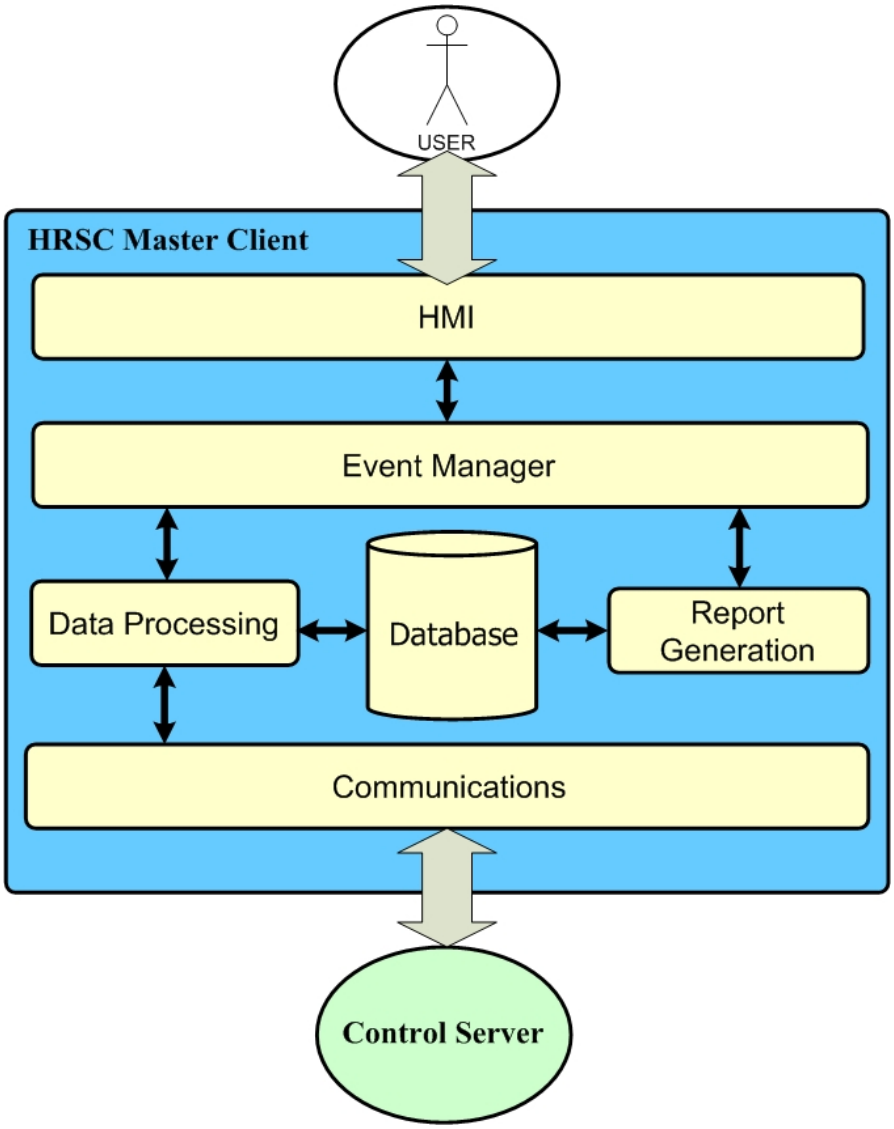


Fig. 6.22: HRSC Client Architecture

The architecture of an HRSC client consists of following basic parts:

- The communicational module that provides the communications between the HRSC client and the Control Server. In cases where the HRSC is a Master Client, the communicational module provides network management in order to establish reliable communication between the robot and another client in the network.

- The data processing module transforms the data input and output data into an adequate format to be stored in the operational database or displayed by the HMI system.
- The operational database stores the sensorial and motion control data as well as the command history. This data is used for further logging and trending into the HMI module.
- The report generation module generates reports on the overall state of the robot based on the historical motion data as well as HMI user command history.
- The event manager is a module that provides the interaction between the data base and HMI interface. It establishes clear rules and priorities for the data flowing into the system.
- The HMI module provides the interaction between the robot and a user. This module is a crucial part of the system and will be discussed in detail below.

As mentioned above, the software architecture allows the use of several HRSC clients running on different remote computers in order to control the robot. In this case, one of the HRSC clients becomes a Master Client and controls network activity and access to the entire system.

6.4.6.3 HMI

The HRSC client system should present the information to the operating user graphically. This means that the operator should be able to see a representation of the humanoid robot being controlled. HMI (Human-Machine Interface) module has been developed to create this representation of the data inside the client system. HMI is the set of methods by which the users interact with the system. The user interface provides representations of [Raymond 2004]:

- Input data, allowing the users to manipulate a system
- Output data, allowing the system to produce the effects of the users' manipulation.

The HMI includes all the functions required for controlling and supervising the functioning of a humanoid robot. The operator using a robot may have to perform:

- Regular process run tasks - Stop and start the walking, operate the controls and make the adjustments required for a regular motion control run and monitor its progress.
- Tasks to deal with unexpected events – Detect abnormal situations and undertake corrective action before the situation disturbs the entire control of a robot. Deal with system failure by stopping motion or implementing a downgraded operation using manual joints motions instead of automatic ones to keep the balance of a system.

At the functional level, the HMI should be compounded of different modules providing the functionality stated below. Figure 6.23 shows the basic modules of the HMI for humanoid robot software system.

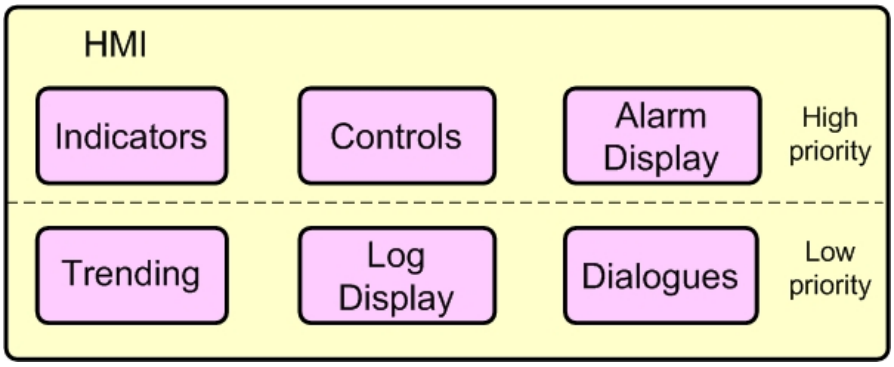


Fig. 6.23: HMI modules

All HMI modules can be divided according to their functionality into high priority modules, which are in continuous execution and low priority modules which are called by a user request.

- Indicators continuously indicate the state of key parameters of a control system. They are, for example, the state of intelligent motion controllers, current position of all joints, etc.
- Controls provide the interaction of a user with a robot. This consists of different sets of pre-programmed commands which can be sent to the robot using HMI in order to modify its current functional state (stop and start biped locomotion, change current motion gait to another, etc.).
- Trending module provides the trending of different parameters of the robot, such as for example the joints velocities, accelerations, currents, body inclinations, forces and torques appears during humanoid robot walking. Real-time and historical trending are possible, although generally not in the same chart.
- Alarm handling is based on limit and status checking and is performed in the Control Server (for example current limit or physical limit of the joint) and then the alarm reports are generated into the HRSC client application and visualized by HMI. More complicated expressions (using arithmetic or logical expressions) can be developed by creating derived parameters on which status or limit checking is then performed.
- Data is logged only if some value changes. Logged data can be transferred to an archive once the log is full. The logged data is time-stamped and can be filtered when viewed by a user. Also it is possible to generate different reports on the humanoid robot state at any time.

Dialogues are used to interact with a user in order to make global changes in the control system of a robot or adjust control parameter. Also, they are used when the user is asked to make a choice on how the system should proceed.

At the application level, the HMI should provide multiple screen support, which can contain combinations of diagrams and text. The whole humanoid robot can be decomposed into “atomic” parameters (e.g. a battery current, its maximum value, its on/off status, etc.) to which a Tag-name is associated. The Tag-names are used to link graphical objects to devices. Standard windows editing facilities should be provided: zooming, re-sizing, scrolling, etc. Online

configuration and customisation of the HMI should be possible for users with the appropriate privileges. Also, it should create links between display pages to navigate from one view to another.

The quality of the HMI design can be measured by the ease with which a user can detect and understand an event and how efficiently he can respond. In every technological solution, the quality of the HMI can predict the acceptance of the entire solution by the intended users. HMI effectiveness is measured by a number of components, such as learnability and productivity. These components are sometimes brought together under the title of "usability", also known as quality of use. Usability is the degree to which the design of a particular user interface takes into account the human psychology and physiology of the users, and makes the process of using the system effective, efficient and satisfying.

6.4.7 Operating System

Since every program should work in the computer system, it is necessary to have a link between software and hardware. This link is provided by an operating system. An operating system (OS) is software that manages computer resources and provides the other programs with an interface to access those resources. An operating system performs basic tasks such as controlling and allocating memory, prioritizing system requests, controlling input and output devices, facilitating computer networking and managing files [Bic 2003].

This thesis work proposes a software control system for humanoid robots which is based on server client technology. As mentioned above, the client is supposed to work at a work station computer, while the server works in the embedded computer system onboard a humanoid robot. Therefore, the requirements for the OS working with Server and Client control applications will be different. (Control Agents' programs are stored in a memory of motion controller and are executed in hard real-time supported by hardware)

Nowadays, there are many OS available in the market. In general, the choice of operating system to be implemented in the control system of a humanoid robot should be based on two criteria - application requirements and compatibility with hardware system.

The HRSC system is a fat client application. It uses a database for storing, processing and visualizing motion control data. It is executed on the conventional work station computer. Moreover, this application doesn't need real time execution of its tasks because it works on the upper level of control where the time requirements are not very strict. Therefore, the choice of the operating system depends on the programmer's choice. But the criterion of an open system allows the programmer to create his own solution for open operational systems such as Linux OS. On the other hand, as mentioned above, the overall software architecture will also remain "free" as in the case of a proprietary OS such as, for example, MS Windows OS will be chosen.

The Control Server application is more complex and demanding. First of all, it will run on the embedded computer inside the humanoid robot. Because of their complex architecture, embedded computers are usually stricter in questions of OS compatibility. Moreover, as mentioned above, the Control Server requires soft real time for performing its control tasks. Therefore, a real-time supporting OS is required. However, in practice it can be shown that the implementation of a non real-time but multitasking system when the embedded computer is provided with fast CPU is enough to provide reliable soft real-time functioning of a Control

Server. The limiting factor here is not the OS, but the bandwidth of communications between Control Server and Control Agents. Experiments, which are presented in following chapter, show the validity of this assumption.

6.5 Communication infrastructure design

6.5.1 Requirements

The communication infrastructure of a humanoid robot is a system that provides the data exchange between different components of a control system. The data exchange in computer systems is usually established by sending digital messages. In order to establish communication links between components of a control system, communication channels must be organized. A communications channel is a pathway over which information can be conveyed. It may be defined by a physical wire that connects communicating devices, or other radiated energy source that have no obvious physical presence (Wi-Fi). Information sent through a communications channel has a source from which the information originates, and a destination to which the information is delivered. Although information originates from a single source, there may be more than one destination, depending upon how many receiving stations are linked to the channel and how much energy the transmitted signal possesses.

In a digital communications channel, the information is represented by individual data bits, which may be encapsulated into multibit message units. A byte, which consists of eight bits, is an example of a message unit that may be conveyed through a digital communications channel. A collection of bytes may itself be grouped into a frame or other higher-level message unit. Such multiple levels of encapsulation facilitate the handling of messages in a complex data communications network. As in every communication system a digital message format is determined by the need for a concrete application, required to make messages understandable for the sender as well as for all the receivers in the communications network.

Therefore, to summarize, the above statements, the communication infrastructure of a humanoid robot can be defined as a system that provides the process of transferring information (digital messages) from a sender to a receiver through the use of a communication channel in which the communicated information is understood by both sender and receiver.

The hardware architecture design provides the physical layer (modules and links) of communication. The software architecture provides functional requirements (data message size, structure, timing, etc.) for communication. The design of communication infrastructure should determine the media (protocol) for data exchange between each module of the motion control system. A communications protocol is the set of standard rules for data representation, signalling, authentication and error detection required to send information over a communications channel.

As it can be seen in figures 6.8 and 6.9 (Hierarchical Hardware architecture and Software architecture for humanoid robot motion control), due to the different functional requirements in control levels, proposed motion control architecture uses two different channels for data exchange. For bottom level communications it is a CAN bus based network and for upper level communication is an Ethernet based network. This section will provide a design of a protocol and communication methods for both of these channels of data exchange.

6.5.2 Organization and Control level communications

The upper level of a control system usually sends and operates big packages of data in order to visualize and monitor the robot's state. As the Ethernet is one of the most common networks for sending large packages of data between computers, the proposed motion control architecture uses an Ethernet based communication network at the upper (Control and Organization) levels and implements client-server technology for software module (HRSC Client and Control Server) interaction (Figure 6.24). In case of wireless communications with a remote workstation, a wireless local area network IEEE 802.11, which links two or more computers without using wires, can be chosen. A wireless Ethernet bridge allows the connection of devices on a wired Ethernet network to a wireless network. The bridge acts as the connection point to the Wireless LAN. This technology allows the use of a remote client and provides the autonomy of a humanoid robot.

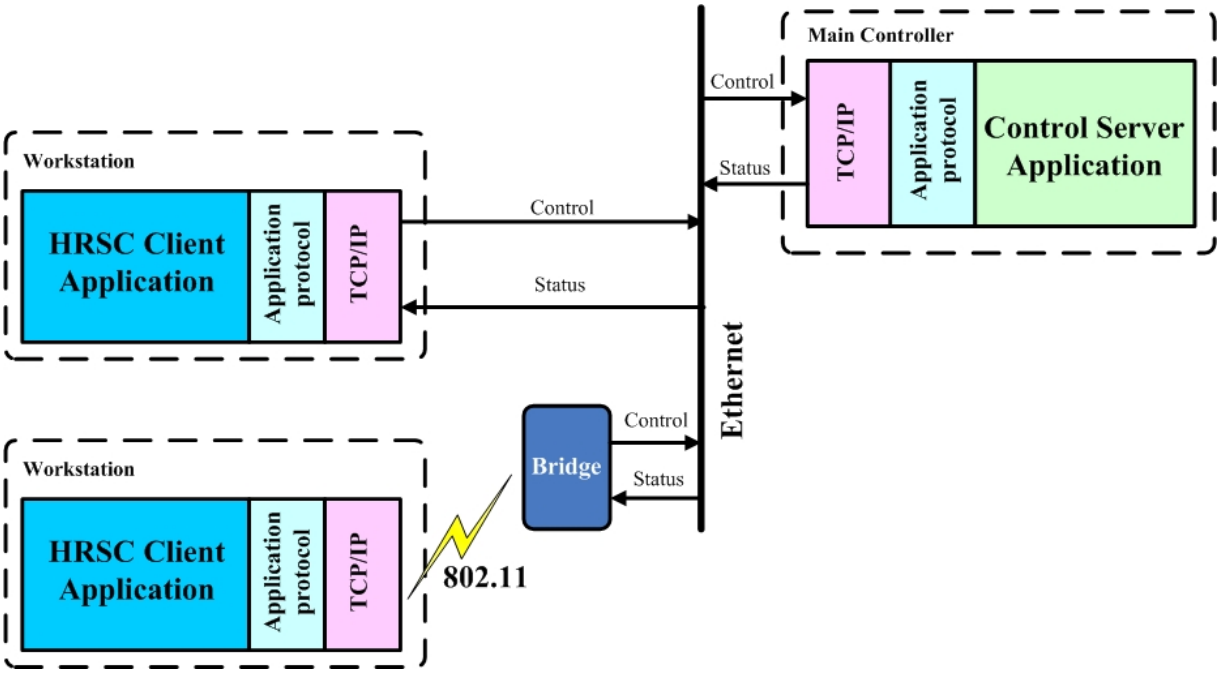


Fig. 6.24 Ethernet based communication system

Ethernet and IEEE 802.11 define a number of standards for the Physical Layer and Data Link Layer, and a common addressing format. However, for reliable data transmission upper layers of communication must be implemented. The TCP/IP Model defines these additional layers [Stevens 94]. TCP/IP defines a set of rules to enable computers to communicate over a common WAN/WLAN network. TCP/IP provides network connectivity by specifying how data should be formatted, addressed, shipped, routed and delivered to the right destinations. The TCP/IP specification defines protocols for different types of communication between computer systems and provides a framework for more detailed standards. TCP/IP is generally described as including the bottom physical layer (Figure 6.25).

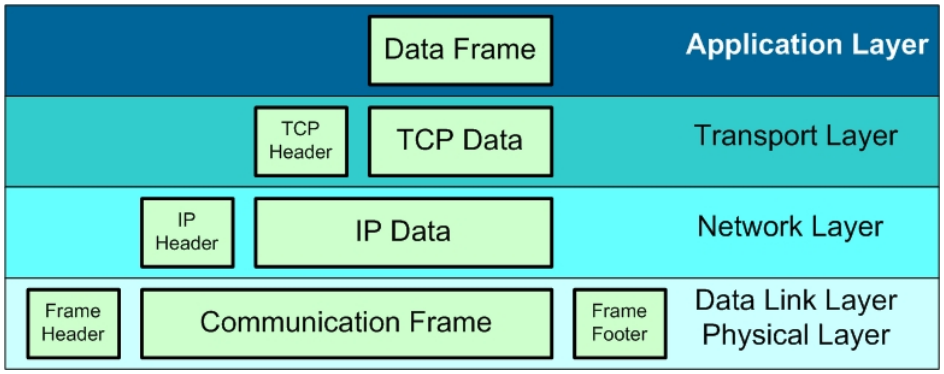


Fig. 6.25: Encapsulation of data frame within a TCP datagram within an IP packet

The layers near the top are logically closer to the user control application, while those near the bottom are logically closer to the physical transmission of the data. Dividing the communication into layers providing or consuming services is a method of abstraction to isolate upper layer protocols from the details related with transmission of bits, collision detection, etc. In the same manner, the lower layers avoid having to know the details of each and every application and its protocol. This abstraction also allows upper layers to provide services that the lower layers cannot, or choose not to, provide [Carpenter 96], [RFC 1122]:

The *Physical layer* is responsible for encoding and transmitting data over network communications media. It operates with data in the form of bits that are sent from the Physical layer of the sending (source) device and received at the Physical layer of the destination device. The Physical layer is considered the domain of many hardware-related network design issues, such as LAN and WLAN technologies.

The *Data Link layer* is the method used to move packets from the Network layer into different hosts. It is responsible for node to node frame delivery. The Data Link layer provides the functional and procedural means to transfer data between network entities and is capable of providing the means to detect and possibly correct errors that may occur in the Physical layer.

The *Network layer* responds to service requests from the transport layer and issues service requests to the Data Link layer. In the Internet protocol (IP) suite, the Network layer performs the basic task of getting packets of data from source to destination. IP can carry data for a number of different upper layer protocols such as TCP.

The *Transport layer's* responsibilities include end-to-end message transfer capabilities independent of the underlying network, along with error control, fragmentation and flow control. It is responsible for delivering data to the appropriate application process on the host computers. The two most widely used transport protocols on the Internet is the connection oriented TCP (Transmission Control Protocol).

Application layer protocol refers to the higher-level protocol used by application programs (HRSC Client and Control Server) for network communication. Data coded according to application layer protocols is encapsulated into transport layer protocol (such as the TCP), which in turn uses lower layer protocols to effect actual data transfer. The common Application layer services provide semantic conversion between associated application processes. Application layer protocols generally treat the transport layer (and lower) protocols as "black boxes" which

provide a stable network connection through which to communicate, although the applications are usually aware of key qualities of the transport layer connection such as the end point IP addresses and port numbers.

For the motion control system for humanoid robots following considerations on the Application level of TCP/IP protocol can be made. The TCP/IP protocol provides technology for data sharing, but only the specific application implements the logic that optimizes performance and makes sense of the data exchange process. When data transmission begins, the sender should packetize (in accordance with Application layer protocol) each piece of data with an ID code that the receiver can use to look up the decoding information. In that way, developed communication protocol hides the TCP/IP implementation details and minimizes network traffic by sending data packages only when they are needed. When a data variable is transmitted by the sender, it is packetized with additional information so it can be received and decoded correctly on the receiving side. Before each data variable is transmitted, a packet is created that includes fields for the Data Size, the Data ID and the data itself. Figure 6.26 shows the Application layer protocol packet format.

Fig. 6.26: The Application layer protocol package format

The Data ID field contains the index of the data array element corresponding to the specified variable. Since the receiving side also has a copy of the data array, it can index it to get the properties (name and type) of the incoming data package. This very effective mechanism is implemented to provide data exchange between the Control Server and different Clients on the Control and Organization levels of the motion control architecture of a humanoid robot.

6.5.3 Device and Sensorial level communications

The Device and Sensorial levels of the proposed architecture provide the autonomous motion and stabilization control of a humanoid robot. This operational mode requires high speed real-time data transmission between the main hardware components and software modules (Control Server and motion Control Agents) of the system. When building control applications of these bottom control levels, communication with the host is often a crucial part of the project. Network nodes (Control Agents) always function as data servers because their primary role is to report information (status, acquired data, analyzed data, etc.) to the host (Control Server) at constant rates. On the other hand, the host replies with control messages (commands and target positions).

As discussed above, the CAN bus provides a robust and deterministic medium allowing decentralized embedded control components to communicate with each other. Therefore, bottom level communications use CAN and CANopen based protocols. The sensorial system of the humanoid robot makes data exchange under the lower CAN protocol and the intelligent motion controllers (device level) uses the upper level CANopen protocol. The same physical layer of these protocols allows them to reside on the same physical network (Figure 6.27).

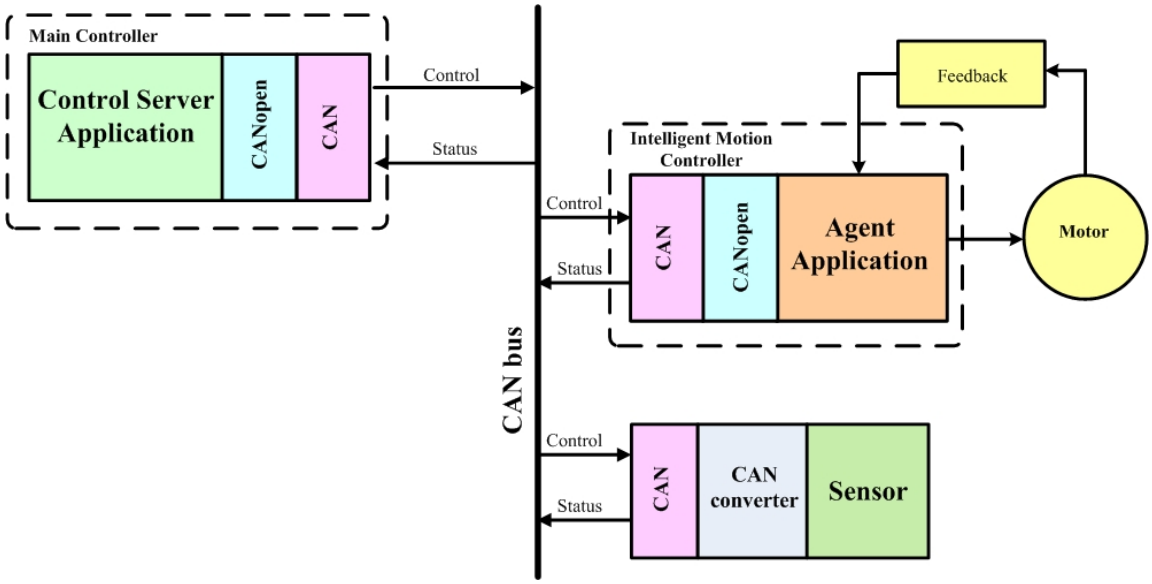


Fig. 6.27: CAN bus based communication system

This communication solution provides data transmission on broadcast type of communication. A sender of information transmits to all devices on the bus. All receiving devices read the message and then decide if it is relevant to them. This guarantees data integrity as all the devices in the communication system use the same information.

In the proposed scheme, a mixed relationship model is implemented:

- A client/server relationship is implemented between the Main controller and intelligent motion controllers (Device level of architecture). The Control Server application acts as a client sending data. The Agent application acts as server that replies with one or more packages containing the requested data or acceptance confirmation.
- A master/slave relationship is implemented between the Main controller and a sensor (Sensorial level of architecture). In this relationship, the Control Server application is designated as the master, which sends or requests data from the slaves (sensors).

Based on levels of abstraction, the structure of the CAN bus based communication infrastructure for the motion control of a humanoid robot can be described in terms of the following layers (CAN Specification 2.0, Part A) (Figure 6.28):

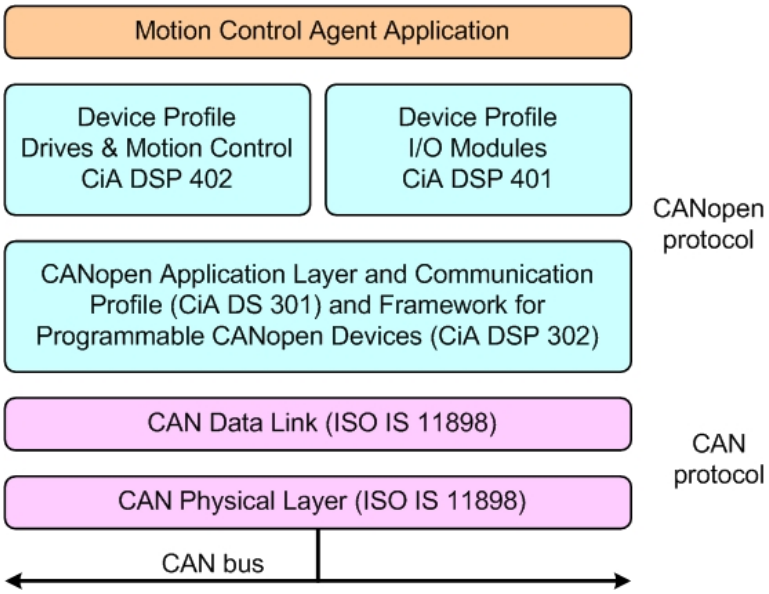


Fig. 6.28: Reference model of a CAN based motion control system for humanoid robots

The *CAN Physical layer* (described in detail in Appendix C) defines how the signals are actually transmitted. Basic tasks include signal level and bit representation. Also this layer defines the transmission medium for the CAN bus network.

The *CAN Data Link layer* provides multi-master capability that means that any CAN node may send a message, if the bus is idle. This layer provides broadcast communication. All messages transmitted are received in all nodes. All receiving nodes decide whether they will accept this message. This guarantees data consistency as all the nodes in the system use the same information. Also, sophisticated error detecting mechanisms and the re-transmission of faulty messages is implemented. This guarantees network-wide data consistency.

The *CANopen Application layer* consists of an addressing scheme and an application layer defined by a device profile. The application layer has support for network management, device monitoring and communication between nodes. CANopen offers several communication objects (COB), which enable system designers to transfer control information, to react to certain error conditions or to influence and control the network behaviours.

A more detailed description of implemented CAN and CANopen protocols can be found in Appendix C. Moreover, as mentioned above, bottom Device and Sensorial levels of communication require a real-time data transmission. Appendix D describes the development of a real-time mechanism in both levels of motion control architecture.

6.6 Conclusions

This chapter addressed the physical level of control architecture for humanoid robots. It proposed the detailed design of open motion software and hardware architectures as well as considerations on the communication infrastructure.

The motion control architecture was designed as an open systems approach. This means that the components, hardware, software, and communication infrastructure are specified in an open manner with specifications that are public. Implemented open software architecture means, above all, the development and implementation of open source software. This permits users to use, change, and improve the software, and to redistribute it in modified or unmodified form. The term open hardware architecture has primarily been used to reflect the use of free/open source software with the hardware and the free release of information regarding the hardware, often including the release of schematics, design, sizes and other information about the hardware implemented in the design of a humanoid robot.

The basic advantage of open architecture is that it promises faster and more economical development of high-quality humanoid robotic platforms that are technologically up-to-date. The proprietary standard solutions such as Windows operating systems and hardware electronics that are parts in the entire system define it as an “open” control architecture with the use of some standard proprietary components.

The proposed hardware architecture fulfils the requirements for open, hierarchical distributed architectures. It is open because it implements the standard hardware solutions and the specification of the architecture is open. It is hierarchical because it is divided into levels of hierarchy (Organization, Control, Device and Sensorial). Also, the proposed architecture provides a high level of modularity and scalability by dividing hardware into modules (systems) specialized in their own control tasks. All the principal modules have been described and their basic design considerations carefully presented.

The developed hardware architecture was designed taking into account the possibility of using conventional electronic components from the automation industry in order to reduce development time and cost and make a flexible and easily upgradeable hardware system.

The structure of the proposed software architecture was designed to offer effective functionality for the motion control for a humanoid robot. It fulfills functional requirements and provides tools to generate, monitor and control bipedal walking. The proposed software architecture is open because it implements standard interfaces. It allows users to create their own motion programs using either the designed functionality described in the developed API or develop their own functionality and programs using the proposed open source code. It makes the system more flexible and easily extensible for future modifications.

The developed communication infrastructure of a humanoid robot provides real-time control data exchange between different components of the architecture. It is a fully open solution using open standard specifications.

Altogether, the proposed open motion control architecture including hardware, software systems and communication infrastructure allows a humanoid robot to carry out synchronized multi-axis dynamical walking, performed by means of real-time control. The physical implementation of the designed architecture with the Rh-1 humanoid platform and some walking experiments will be presented in the following chapter.

Chapter 7

Implementation of the motion control architecture with different humanoid platforms and walking experiments

CHAPTER 7

Implementation of the motion control architecture with different humanoid platforms and walking experiments

7.1 Introduction

Chapter 3 presented the conceptual study of the motion of humanoid robots. Chapters 4 and 5 provided the development of principal motion control algorithms – joint control and stabilization control of a humanoid robot. Chapter 6 discussed the detailed design of hardware and software architectures for humanoids. This chapter will present the implementation of developed architecture and control algorithms with two different robotics platforms – Rh-1 and HRP-2 humanoid robots.

The implementation of control architecture is the materialization of the ideas on control, hardware and software presented in previous chapters as physical electronic components and software programs in order to obtain a fully operable system – a humanoid robot capable of walking. Therefore, the software and hardware architectures presented above were implemented with an Rh-1 humanoid robot – a research platform built in the Robotics Lab of the University Carlos III of Madrid.

The creation of hardware architecture - basic components implemented inside the Rh-1 robot and its physical distribution and linking will be presented. Implementation of the software architecture is done by a program containing both the graphical interface and the internal motion control algorithms. The latter were presented in previous chapters. This chapter presents the basic functionality of the developed software in terms of HMI for operating the Rh-1 robot and adjusting its motion control parameters.

Another aspect of the control architecture is the realization of the motion control algorithms. The Joint control algorithm was implemented and tested with the Rh-1 humanoid robot. Stabilization control was implemented with the HRP-2 humanoid robot.

In order to introduce these test platforms and clarify further experimental parts, this chapter will present mechanical design and characteristics of both platforms.

Finally, the experiments that will be presented in this chapter have as a goal to prove the applicability and reliability of the designed open motion control architecture and motion control algorithms for humanoid robots.

7.2 Humanoid Robot Rh-1

7.2.1 Considerations for the humanoid robot Rh-1 design

Since industrial robots cannot be easily adapted to assist human activities in every day environments such as in hospitals, homes, or offices, a need for the robots that could interact with a person in a human-like manner is growing steadily. Wheel robots sometimes cannot be used in these kinds of environments, because of the obvious restrictions posed by the use of the wheels. For example, it is impossible for these kinds of robots to go up and down stairs, or to clear obstacles on the floor. Therefore, the humanoid robots are expected to play a more important role in the future.

As was already discussed in the chapter 1, one of the most exciting challenges that has faced the engineering community in the last decades was to get a machine of a form similar to a humanoid robot that could do the same activities as a human being, in addition to walking in a similar manner.

There are several reasons to construct a robot with these characteristics. Humanoid robots will work in a human environment with greater effectiveness than any other type of robot, because the great majority of environments where it would interact reciprocally with a human are constructed taking into account the dimensions of the latter. If it is assumed that a machine should complete dangerous tasks or work in extreme conditions, in the ideal case its anthropometric measurements must be as close as possible to the ones of their prototype. Inclusively, there are professionals who adduce that for a human being to interact naturally with a machine, it must look like him. All these reasons have had an impact on the development of the Rh-1 humanoid robot.

Rh-1 is a new humanoid robot in the second phase of the Rh project. The robot is developed for several principal tasks such as human care, operation in dangerous for human health industrial conditions, entertainment etc. The Rh-1 is designed and constructed in order to fulfil several conditions. The robot has to be able to:

- walk straight and turn its body on the plane surface
- go upstairs and downstairs
- simulate human hand movements
- gesticulate with its hands (give signals, salute)
- take and manipulate objects of up to 0.5 kg

The main design concept of Rh-1 robot is a lightweight and compact (comparable with a human) size. As a result the robot has 1200 mm height (without the head) and 50 kg weight.

7.2.2 Mechanical design of Rh-1

The basic configuration of the Rh-1 humanoid robot, possesses an anthropomorphic design, and is equipped with two legs, each one with 6 rotational degrees of freedom (DOF). In addition, three of these generate movements in the sagittal plane (lower hip, rolls and ankle), two in the frontal plane (ankle and middle hip) and one in the cross-sectional plane (upper hip). The robot has the trunk where the hardware components will be located, and it has two arms. Mechanically speaking, there are 9 degrees of freedom in the upper part of the robot, the trunk has 1 DOF in the neck and in the arms there are 4 DOF which are distributed in the following ways: 2 DOF in the shoulder, 1 DOF in the elbow and the last 1 DOF in the wrist, as can be seen in the Figure 7.1.

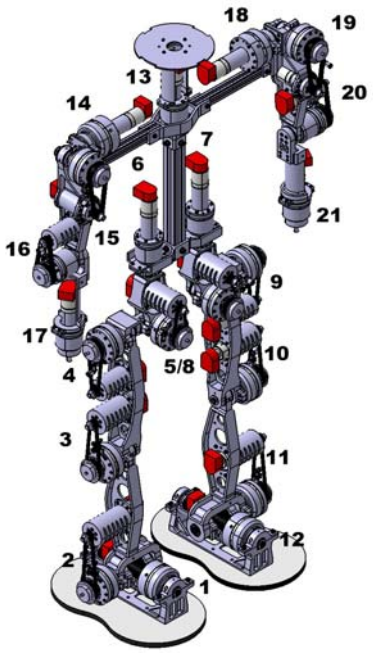


Fig. 7.1: Mechanical configuration of the Rh-1 robot

Further experiments proved that this configuration allows the humanoid to walk efficiently.

The design parameters require careful consideration and planning to secure walking stability. A degree of freedom in the frontal plane had to be put in the trunk, which will be used to stabilize the robot while walking.

Table 7.1 shows the specification of each articulation and weight distribution of Rh-0.

	D.O.F. Number	Axis of movement
LEGS	12 D.O.F.	
Hip Knee Ankle	3 (x2) 1 (x2) 2 (x2)	Pitch, Roll, Yaw Pitch Pitch, Roll
ARMS	8 D.O.F.	
Shoulder Elbow Wrist	2 (x2) 1 (x2) 1 (x2)	Pitch, Roll Pitch Yaw
TRUNK	1 D.O.F.	Yaw
HEAD	2 D.O.F.	
Camera	2	Yaw (left – right), Pitch (up – down)
TOTAL:	21 D.O.F. (23 D.O.F. with the camera)	
Weight:	Mechanical structure: 39 kg Batteries: 6 kg Control Electronics: 5 kg Cabling: 2 kg	
	Total: 52 kg	

Table 7.1: Specifications of the Rh-1 humanoid robot

Figure 7.2(a) shows the prototype created in a CAD/CAM program that has already been finalized, previous to its entrance to the factory, after adjusting to the maximum all the constructive and engineering parameters that were considered. Figure 7.2(b) demonstrates the first prototype of the Rh-0, mounted and equipped with motors and gear systems.

a) b)

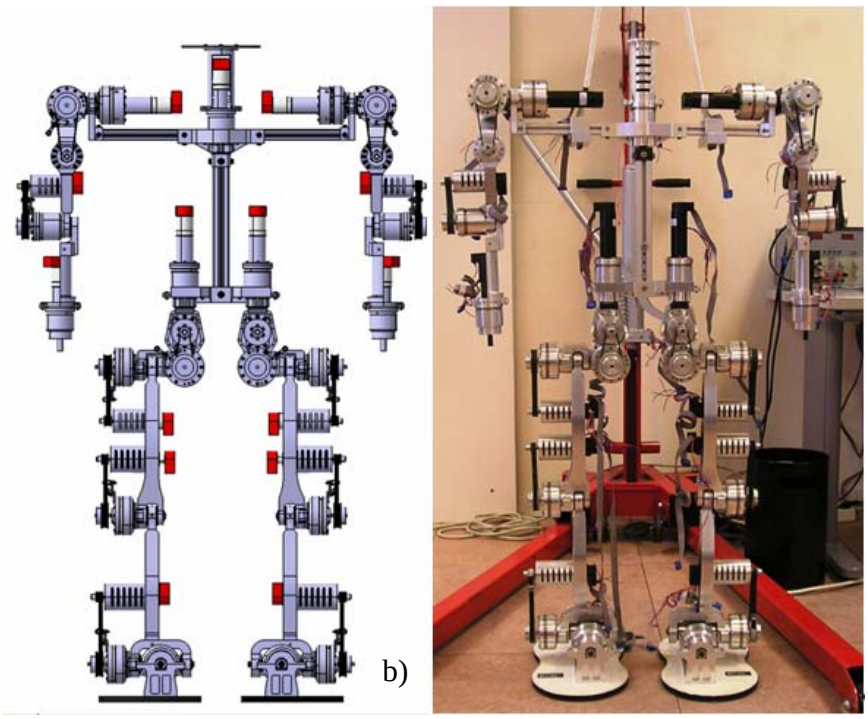


Fig. 7.2: Development of the Rh-1: a) Development in CAD/CAM and b) Final Prototype

Finally, the mechanical structure of the robot was developed and constructed afterwards.

7.2.3 Hardware architecture implementation

The main parts of the hardware of the contemporary existing humanoid robots are the "custom-built" components. The use of specialized and single-purposed components in hardware architecture reduces functionality and scalability of the architecture. Also, the time to place the system in operation rises. Thus, it implies the increase in usage of some technologies from the industrial automation field in humanoid robotics due to its affordability and reliability.

The designed control architecture of the Rh-1 robot was implemented using conventional electronic components of the automation industry in order to reduce the development time and cost and to have a flexible and easily upgradeable hardware system.

The hardware architecture for the humanoid robot has some important restrictions posed by limited availability of space. As was discussed in the previous chapter, the open motion control architecture for humanoid robots is designed as a distributed control system.

After all hardware components and modules were defined in the pervious chapter, the physical distribution of components inside the humanoid robot Rh-1 body was made (Figure 7.3)

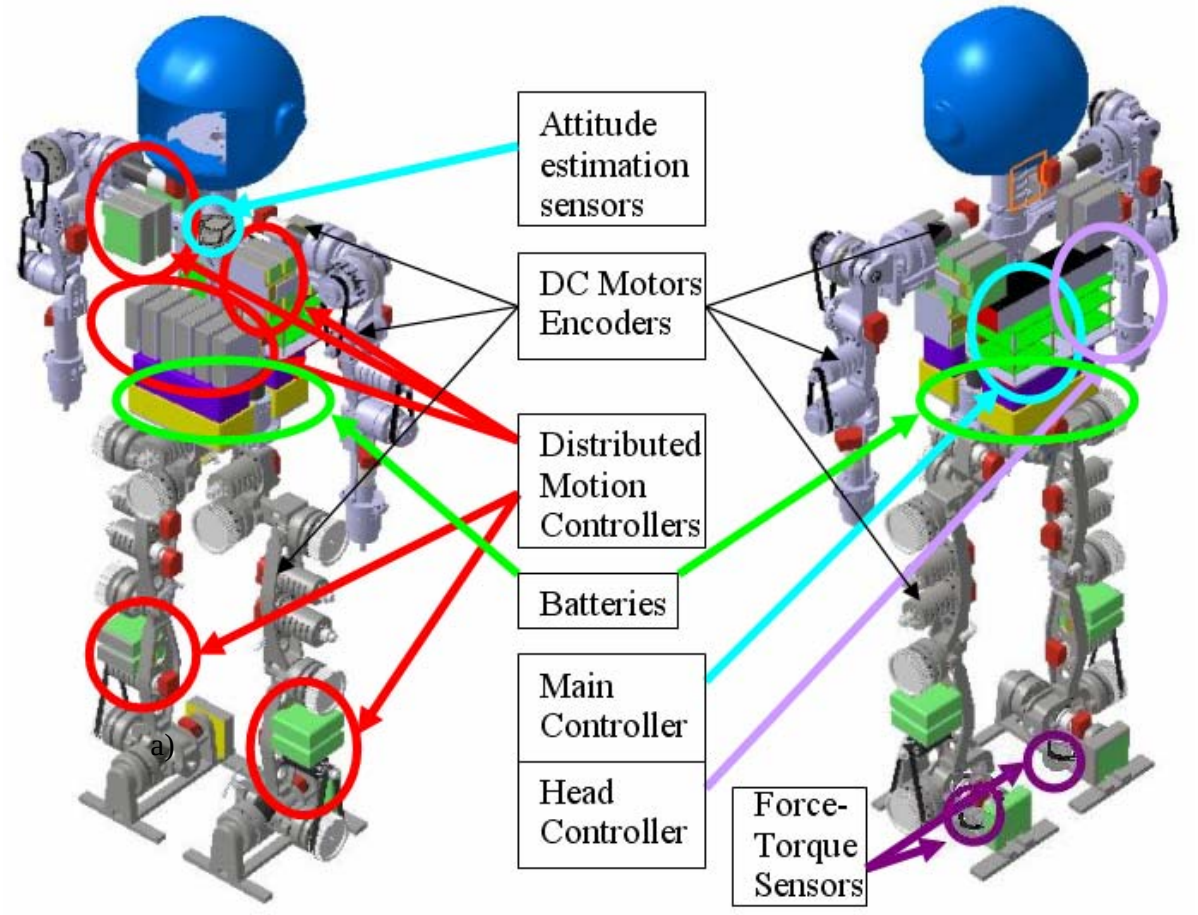


Fig. 7.3: Hardware distribution inside the humanoid robot

Basic considerations taken into account for hardware components distribution were:

- Electrical motor should be located near the joint's rotation axis
- Intelligent motion controllers have to be distributed as close as possible to the motor in order to reduce cabling effort
- Place the Main Controller (as well as the Head Controller) into the backpack of the robot and provide good air flow and ventilation conditions for their CPUs.
- Place attitude sensors in the trunk and force-torque sensors in the feet of the robot
- As the battery is one of the heaviest hardware elements in the hardware architecture, place it in the waist of the robot

The distribution of electronics shown in Figure 7.3 provides easy access to every component for cabling, service and maintenance. The overall weight is distributed symmetrically in order to place the COG of the robot in the trunk and approximately at waist level. And finally, the hardware distribution was made taking into account the anthropometrical structure of the humanoid robot Rh-1 and allows for easy design of the cover.

Once all hardware components are distributed inside the humanoid's robot body, the next step is to make a reliable connection between them. The electrical signals between the devices inside the control system are transmitted using cables. Cabling of the control system such as a humanoid robot having a lot of electrical motors, sensors and other control devices is a complex task.

As a humanoid has a lot of joints moving one in respect to another, the cabling system should be designed in a way that does not disturb the motion of the robot's joints. Moreover, in order to reduce the real cabling efforts and make the design of a cover, the cabling of the humanoid robot Rh-1 was first made using a CAD program. Figure 7.4(a) shows the cabling design of the knee joint. Figure 7.4(b) provides the cabling design of the torso of the Rh-1 robot.

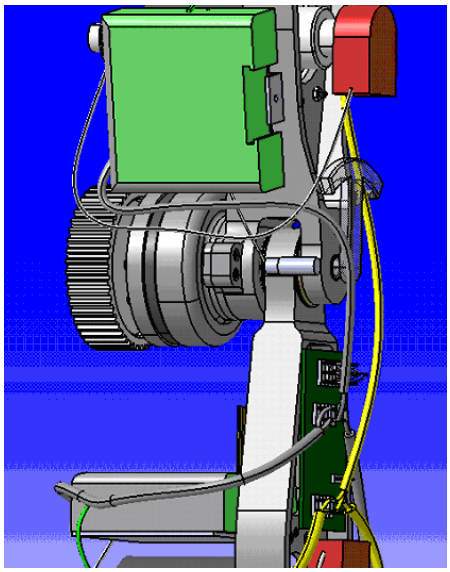
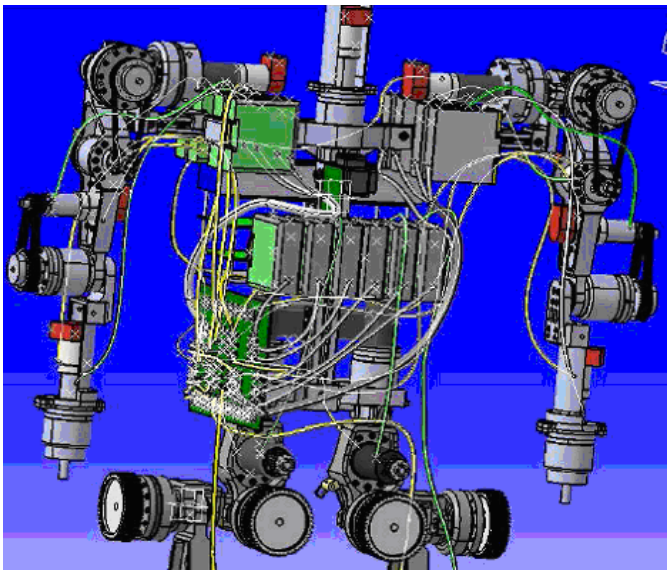
a)  b) 

Fig. 7.4: a) Knee joint cabling design b) Torso cabling design

After the entire design of the hardware components and cabling system was accomplished, the anthropomorphic cover was designed and produced. Figure 7.5 shows the covered structure of the humanoid robot Rh-1 and its practical realization.

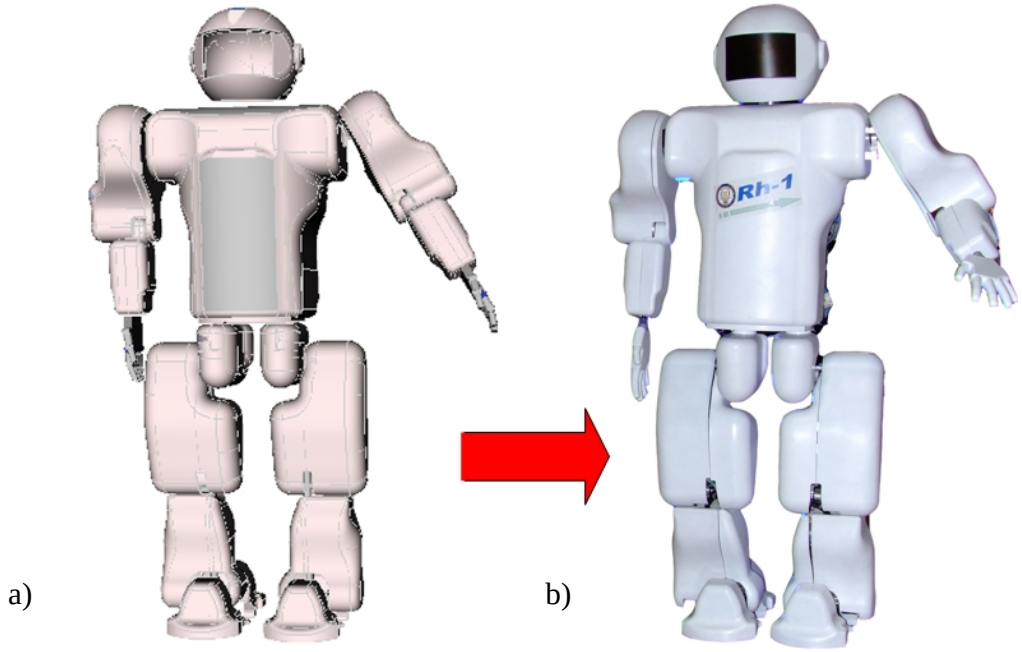


Fig. 7.5: Humanoid robot Rh-1 a) Cover design b) Realization

Finally, Rh-1 is the humanoid robot built as an investigation platform to develop algorithms of a stable biped walking. Once the first stages (mechanical and hardware systems design) were accomplished successfully, the work centred on software architecture implementation allowing for the stable biped locomotion and simple interaction of the Rh-1 humanoid robot with humans and the typical environment.

7.2.4 Software architectures implementation

According to the software architecture proposed in the previous chapter, three basic software modules were implemented with the humanoid robot Rh-1:

- Control Server – program running on onboard Main Controller of the robot and providing general control and communication tasks of the robot
- Control Agent- program running in the intelligent motion controller and providing local motion control tasks (joint) control
- HRSC Client – program running on the remote desktop and providing user interface and motion data visualization and storing

All software code for Rh-1 humanoid robot motion control system was implemented using C-based programming languages. C-based programming language is regarded as a middle-level language, as it comprises a combination of both high-level and low-level language features.

Thus, it is useful for both, for high-level components programming (Client software, user interfaces) as well as for bottom level components programming as Control Server and Control Agents.

All internal functionality of these modules was presented in the previous chapter. Therefore, this chapter will present only the realization of user interfaces.

The Control Server program runs inside the onboard main control system of the humanoid robot Rh-1. As was discussed in the previous chapter, it receives the data that arrives from the client program and provides all upper level control tasks: sensorial data acquisition computes stabilization control, modifies motion patterns and sends it to control Agents of every joint of the robot. As the Control Server is a real-time application working with low level hardware and communication devices (Ethernet, CAN), it was implemented using C programming language, because C was designed especially as a system implementation language and has very good low-level capabilities.

The Control Server is realized with the console terminal application (Figure 7.6). It uses a simple Command Line Interface (CLI) as a mechanism for interacting with the program by typing commands to conduct the system. A CLI of the Control Server can generally be considered as consisting of syntax and semantics. The syntax is the grammar (set of rules) that all commands must follow. The semantics define what sort of operations are possible, and on what sort of data these operations can be performed.

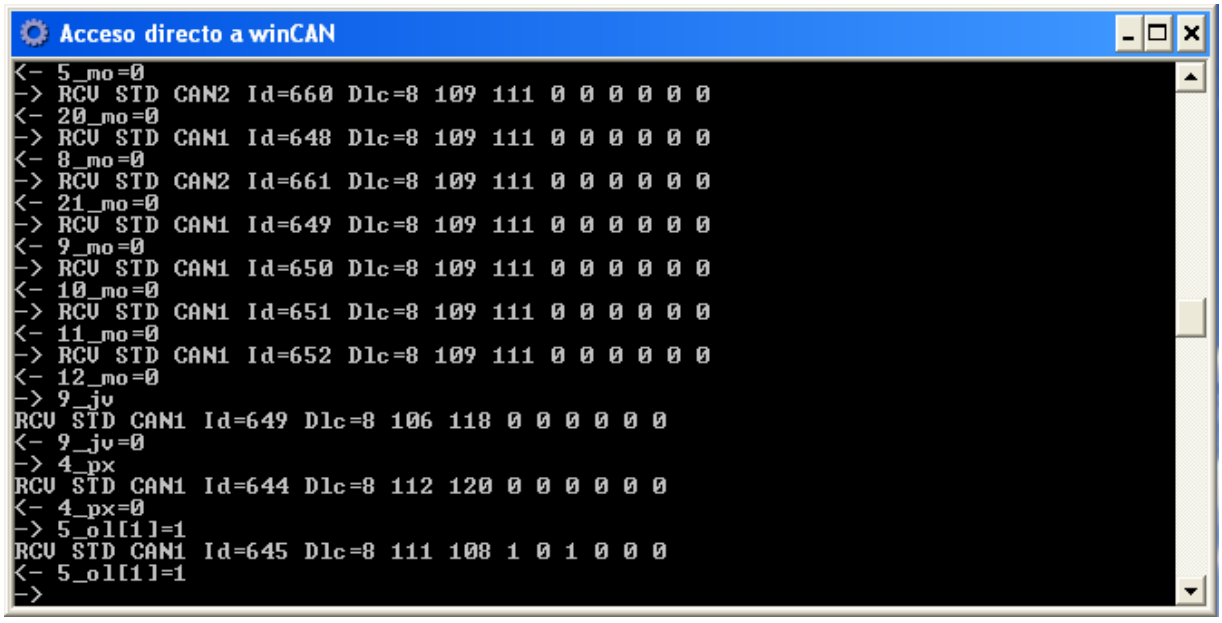


Fig. 7.6: Control Server Terminal screenshot

The terminal is provided with the parser, which analyzes a sequence of tokens introduced by the user to determine the grammatical structure of the introduced command with respect to a given formal grammar of pre-defined commands. The parser also checks for syntax errors in introduced commands. When the introduced command is correct, it is executed and the Server

responds with a reply message that contains the result of the requested operation or notification of starting or finishing some control process.

The main tasks of the control terminal program are to provide access from to the Control Server application for the adjusting of motion control parameters and sending some direct commands to the robot's intelligent joint motion controllers.

The Control Server program runs in the remote workstation and provides the user interface for Rh-1 humanoid robot. It is the program that controls the system and assesses the state of the system. It presents the graphical and textual information to the user, and the control sequences the user employs to control the humanoid robot Rh-1. HRSC client is a Windows based application created using high-level object oriented C++ programming language.

It is Multiple Document Interface (MDI) application. It has a resizable general-purpose window and different dialog frames reside under a parent window. Children frames are designed for control and monitoring of different tasks (Figure 7.7)

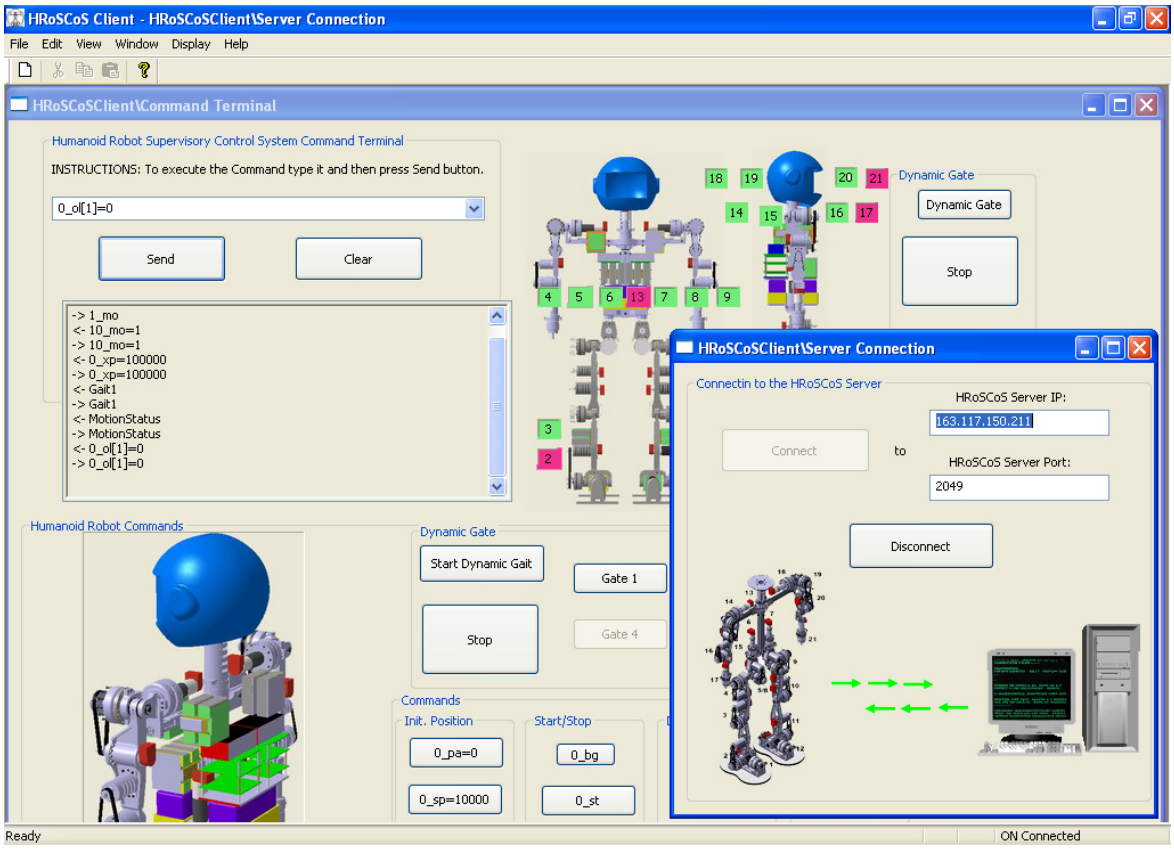


Fig. 7.7: HRSC client screenshot

As it was described in the previous chapter, the HRSC client for humanoid robot Rh-1 provides different indicators on the robot's state (for example, intelligent motion controller on/off indicator). Also it provides different controls for changing actual parameters of some variables and sending pre-programmed commands to the robot. In addition, it is possible to

enable a terminal mode under the HRSC client application. It will replicate all the Control Server's processes and internal commands in the remote terminal window. Furthermore, it is possible to choose some parameters and organize its trending online or perform a logging of parameters in a parameter's data base. Alarms handling provides the graphical indication of unexpected situations. Implemented dialogues allow interacting with a user in order to make global changes in the control system of a robot or make control parameters adjustments. The access to the parameters of the Control Agents is organized also with the implementation of dialogs in the HRSC client application. Moreover, different dialogs are used when the user is asked to make a choice on the way the system should proceed.

As it was described above, the developed software system, besides the user interface, provides motion control algorithms allowing the robot to make motions. The next section will present experiments carried out with the Rh-1 humanoid robot provided with developed hardware and software systems.

7.2.5 Walking experiments

In order to prove the applicability of the discussed architecture, different walking experiments were carried out. The goal of these experiments was to show that the humanoid robot Rh1 provided with the open motion control architecture could walk stably. Another goal was to study the reliability of hardware and software components as well as different control algorithms.

The experiments carried out tested different types of motion (gaits) of the humanoid robot in the open-loop mode (without implementation of stabilization control). First of all, in order to prove the mechanical structure and motors of the Rh-1 robot, the one foot balancing experiment (figure 7.8) was provided. In this experiment, the robot is maintained on one foot and makes cyclical squatting (up and down) movements. After this experiment proved the good mechanical characteristics of the robot, other walking experiments were started.

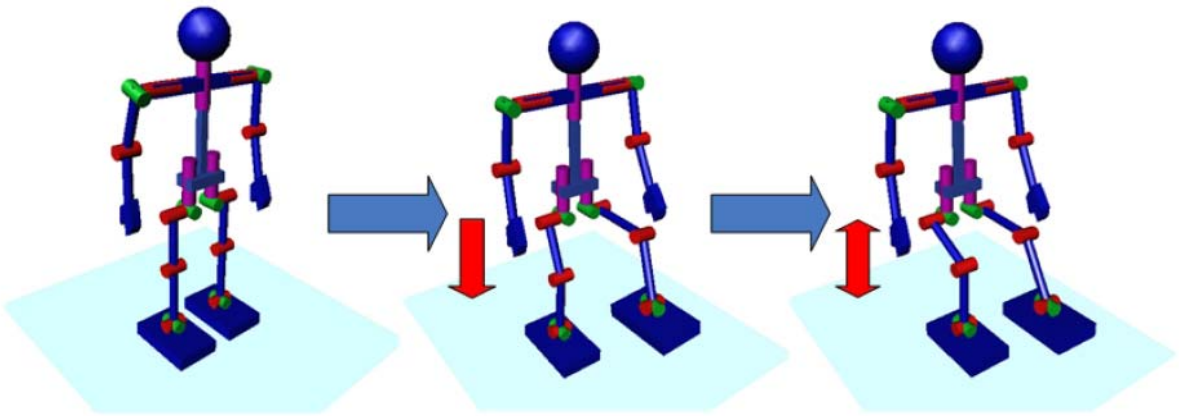


Fig. 7.8: One foot balancing experiment scheme

The main walking experiment evidently is the forward motion gait (figure 7.9(a)). For this motion, the robot lifts one foot and translates it to the next stepping position in front, moving the whole body forward. The robot moves the centre of its mass from the supporting leg to the stepping leg. The cyclical execution of these motions produces the forward walking gait.

Backward gait (Figure 7.9(b)) sequence is realized in the same way as forward locomotion. The difference is that the foot in the air as well as the robot's body is transferred backwards. In the lateral gait motion (Figure 7.9(c)) the foot and the body are transferred in a lateral direction, and in the turning gait (Figure 7.9(d)), the robot executes forward locomotion with simultaneous turning of the support foot that results in turning of the entire body.

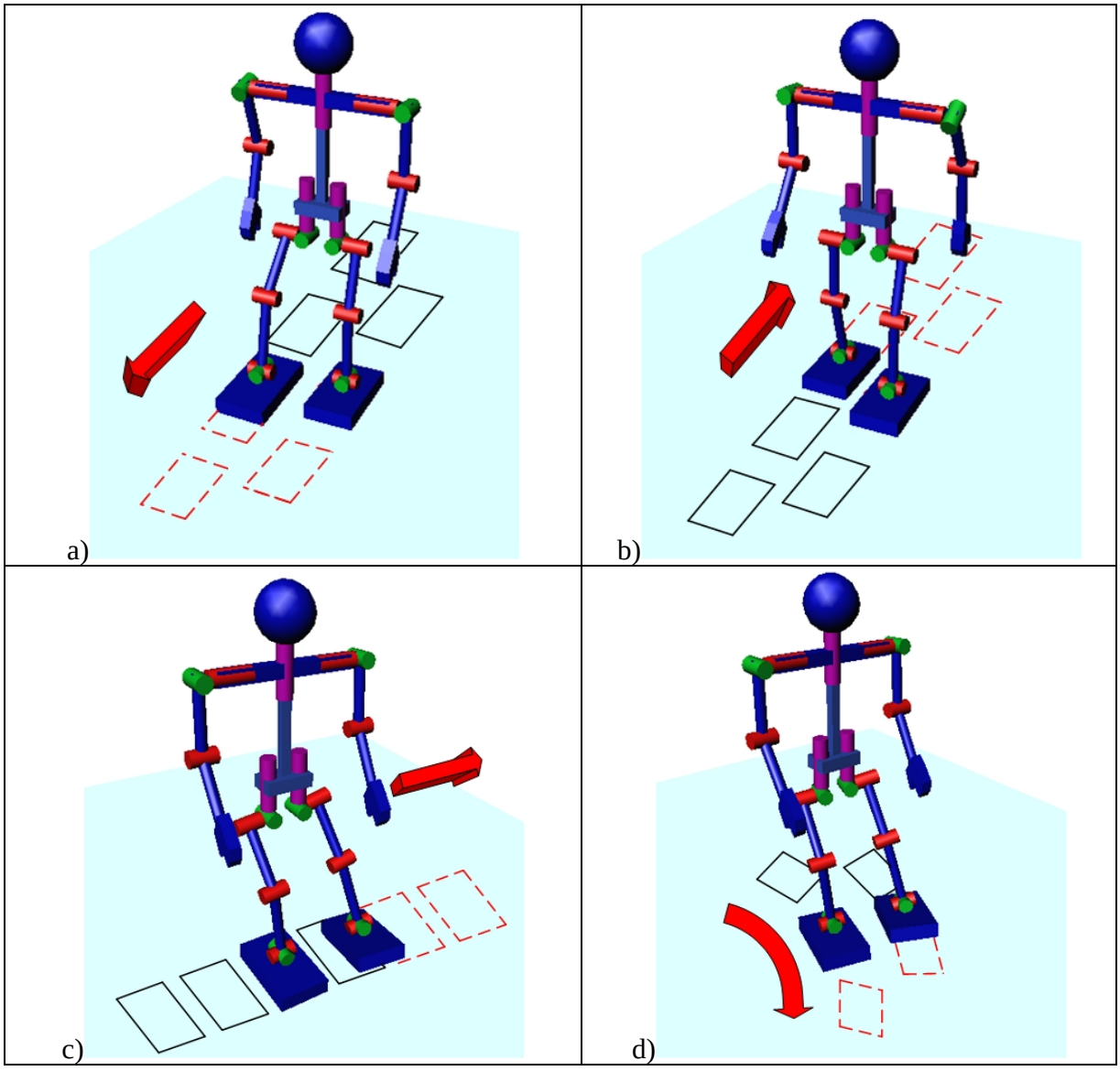


Fig. 7.9: Rh-1 walking experiments scheme a) forward gait b) backward gait c) lateral gait d) turning gait

The main characteristic showing the quality of motion control architecture design is the stable locomotion of the robot in the dynamic forward walking gait, which is the fundamental one for achieving walking stability. Figure 7.10 shows the snapshots of the provided experiment. Although, in this experiment the dynamic motion pattern was executed, the motion is still remain open-loop, because the stabilization control is not implemented yet with Rh-1 humanoid robot.

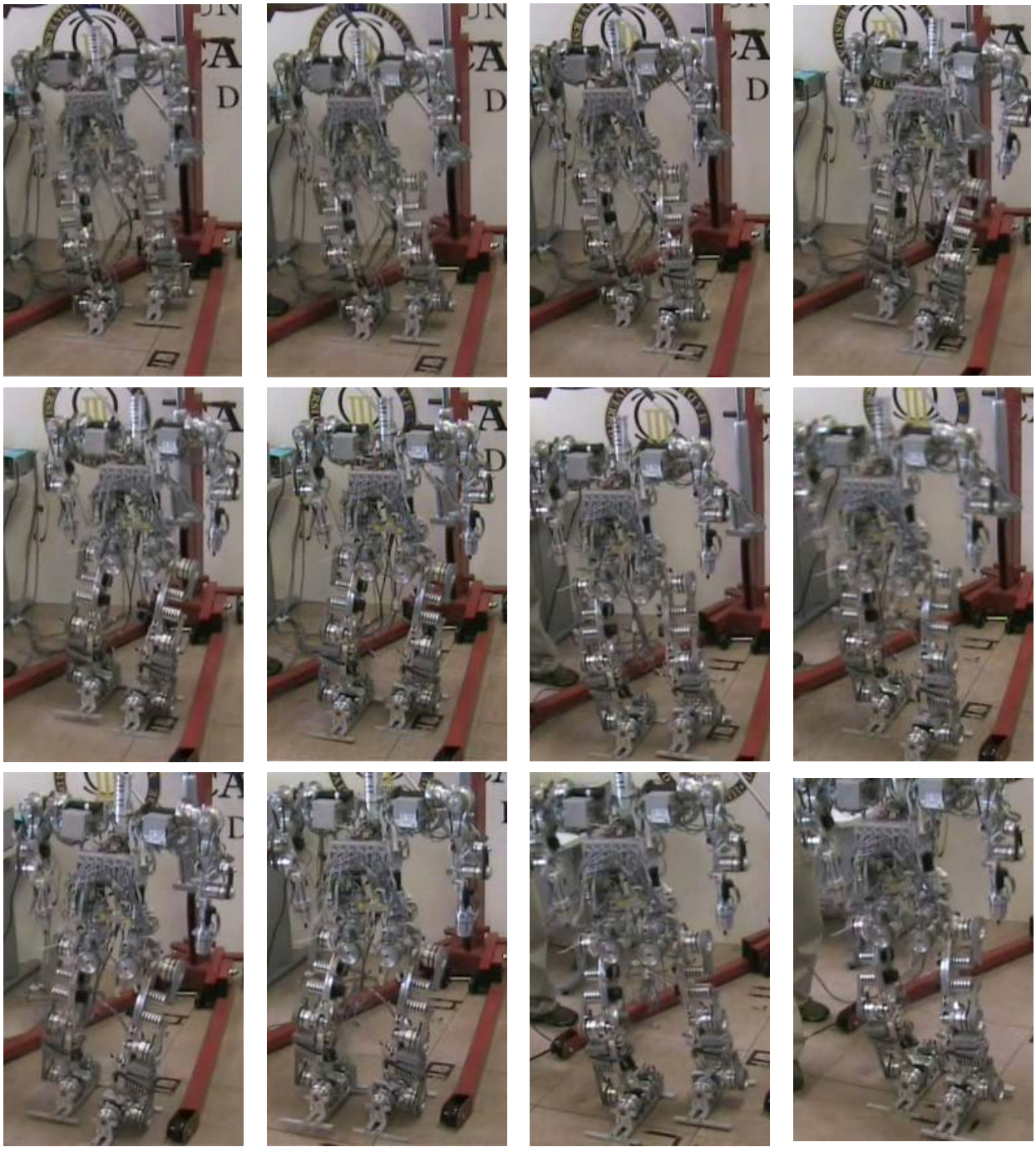


Fig. 7.10: Snapshots of forward walking gait

The robot in this walking experiment is provided with a control hardware and software system, but it is not totally autonomous because the power control system is not installed. The power is provided from an external source by the cable. Moreover, in these experiments the robot operates without a cover. It allows for observing the functioning of the mechanical and hardware systems and making their adjustments.

As it was mentioned above, the main characteristic showing the quality of motion control architecture design is the biped locomotion of the robot. Snapshots of the robots walking prove that a humanoid robot Rh-1 provided with developed motion control architecture can walk. Moreover, the auxiliary characteristic for evaluating the control system is the actual joint position variation compared to the target joint position. The following graphs show the position variation of each of the 12 joints of the Rh-1 robot involved in biped motion.

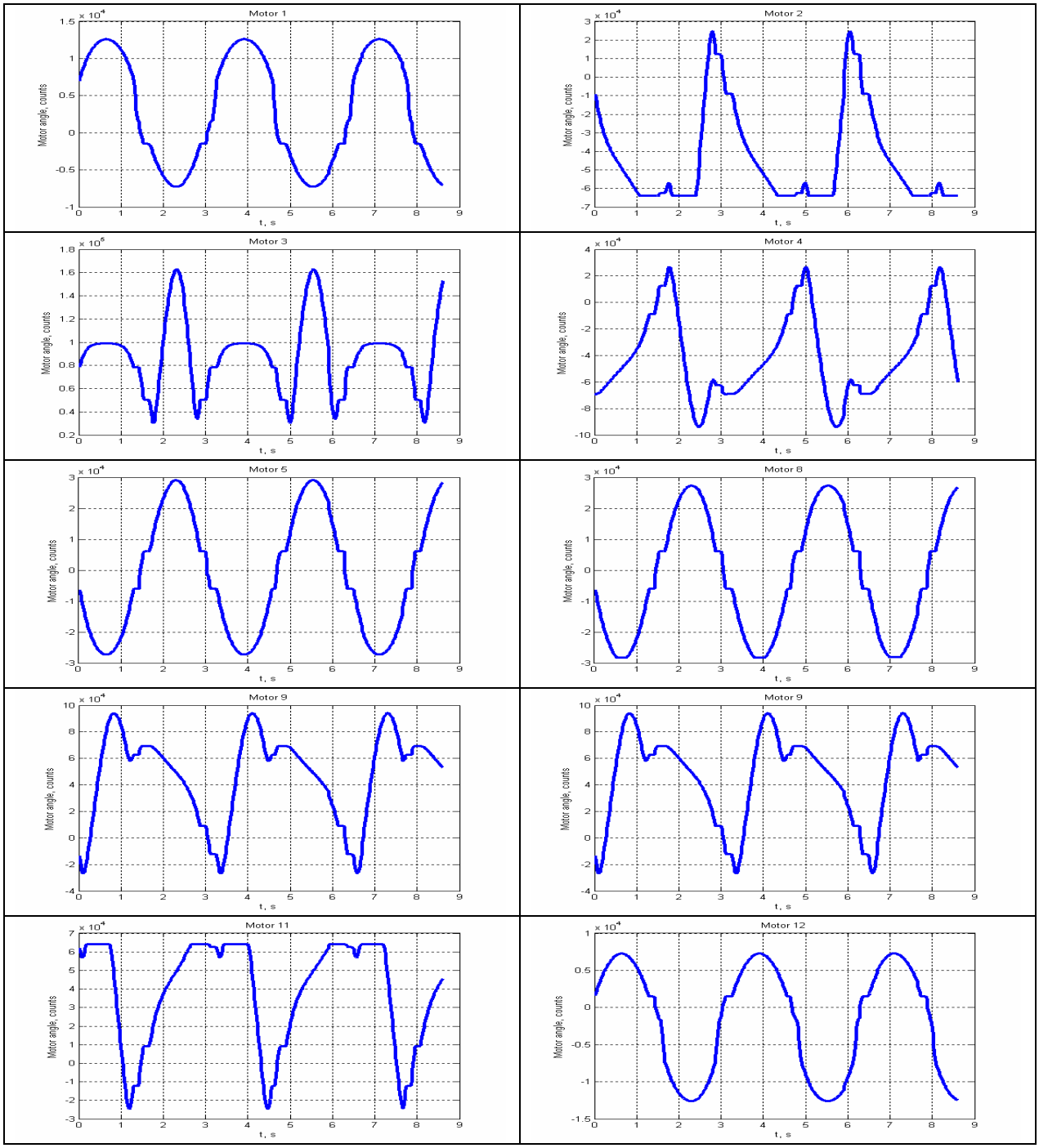
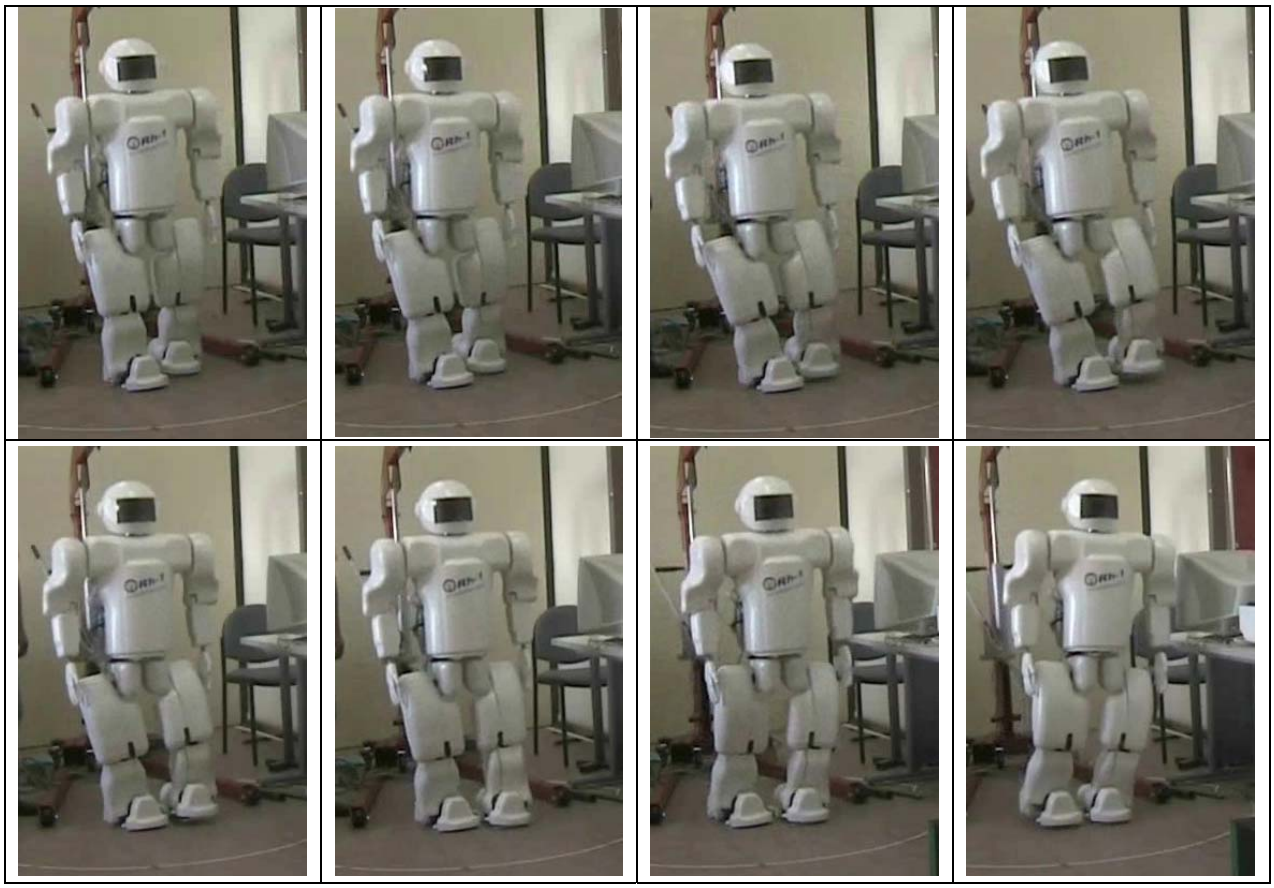


Fig. 7.11: Joint position variation

It can be observed, that the motor position of all joints exactly follows the target position proposed by the motion pattern. This proves the adequacy of the joint control algorithms implemented in the robot. Nevertheless, as can be observed from the snapshots of the experiment, the motion remains not very stable because of the open-loop control scheme. The further implementation of the proposed stabilization control algorithm explained in the chapter 5 could radically improve the Rh-1 humanoid robot walking.

After the first experiments proved the applicability and reliability of the proposed hardware and software, the cover was designed, the power control system was mounted into the robot, and the next series of walking experiments was carried out. Figure 7.12 shows snapshots of the provided experiments. In these experiments, the Rh-1 humanoid robot walks in autonomous mode with all control hardware onboard. The communication between the Control Server and remote HRSC client program is wireless.



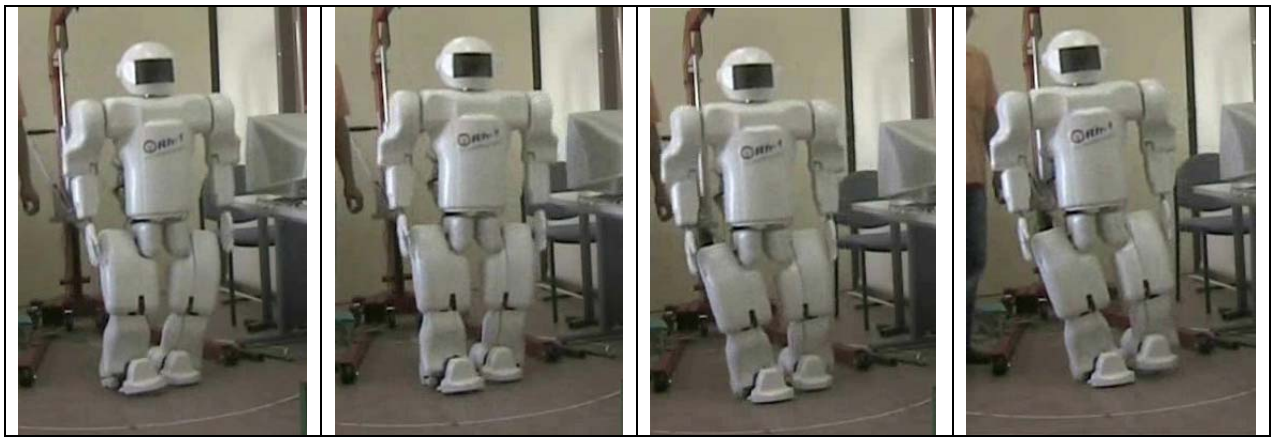
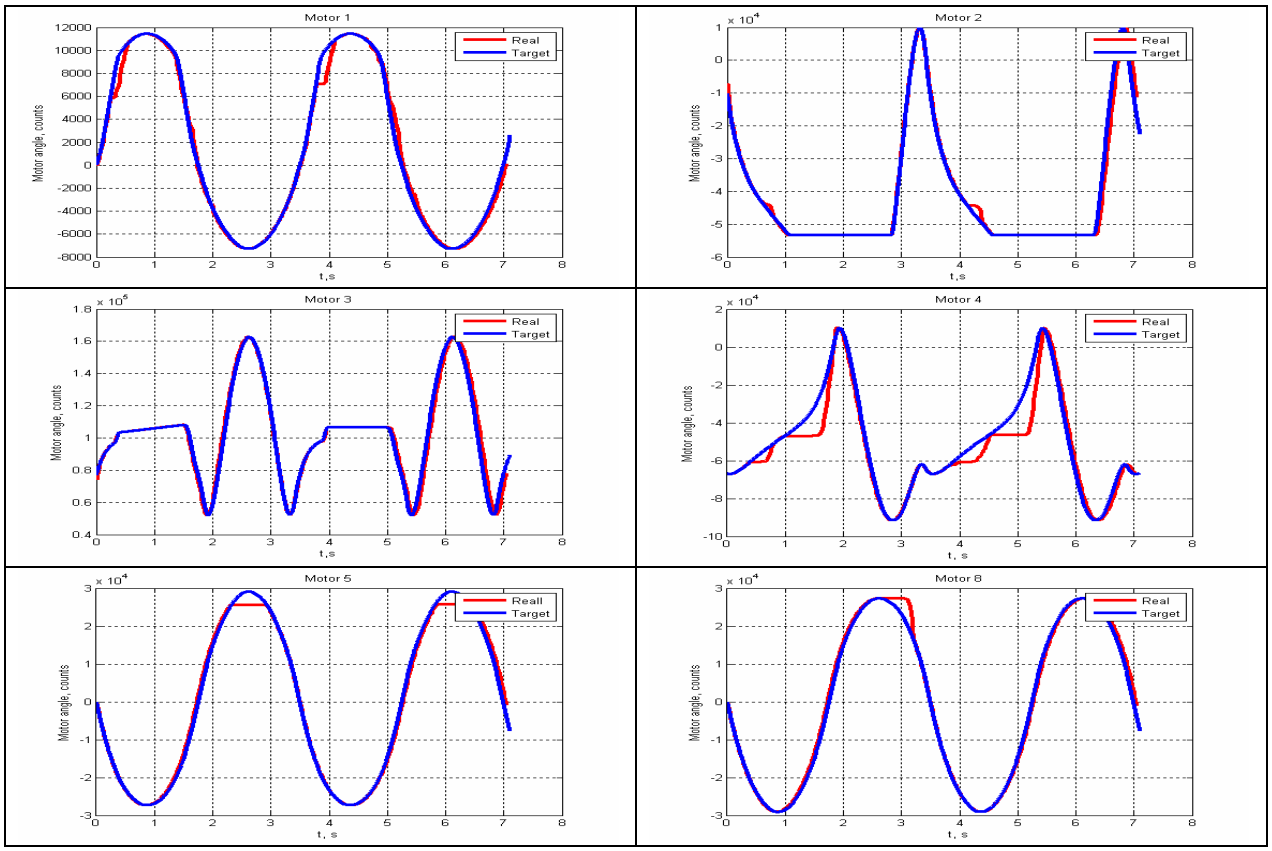


Fig. 7.12: Snapshots of the humanoid robot Rh-1 walking

In these experiments the robot walked worse. Although practically the same motion patterns were implemented, the walk was less stable. The following graphs show the comparison of variation of the real motors' position and that of the target one proposed by the previously generated motion pattern.



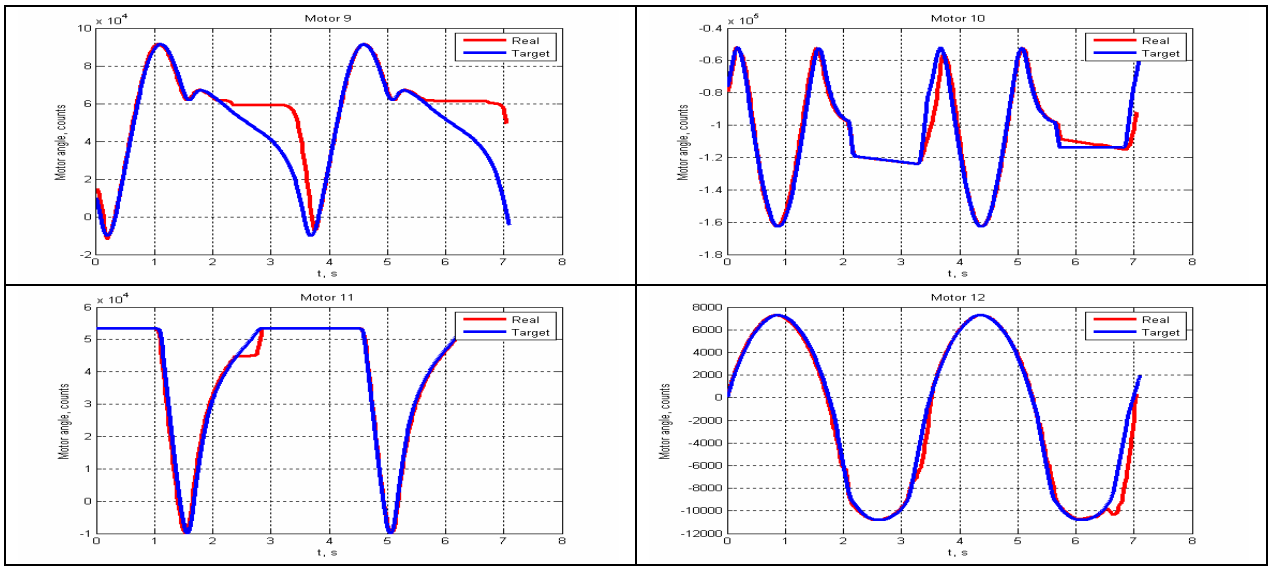
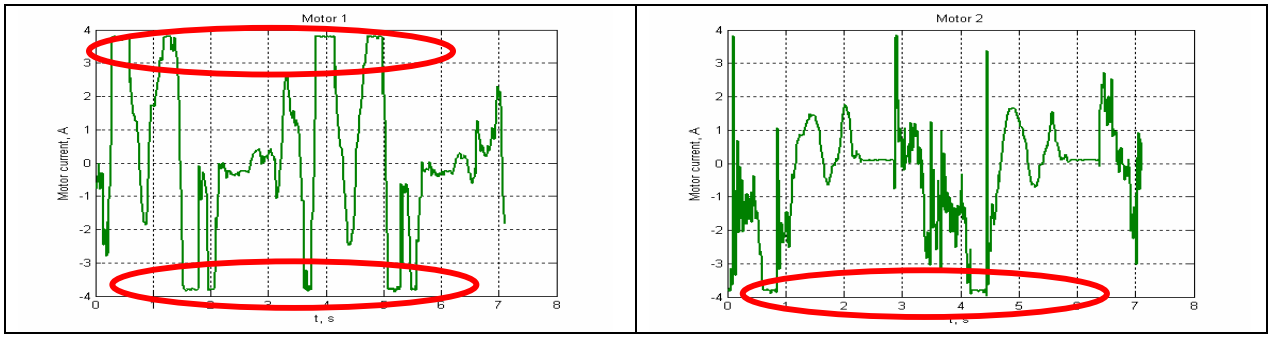


Fig. 7.13: Joint position variation (actual vs. target)

It can be observed, that in this case, some joints didn't follow the target motion pattern. The worst functioning can be noted at the robot's hip joints (motors 4 and 9). The fact that the real and target trajectories do not coincide confirms the worst walking. On the other hand, it can be shown that this problem is not caused by the designed motion control system. Implemented hardware and software systems, as well as joint control algorithm, work stably and don't bring these errors into the robot's walk.

In order to find the cause of these walking problems suffered by the Rh-1 robot, let us consider the graphs of the actual current variation of all the joints involved into the biped motion (Figure 7.14).



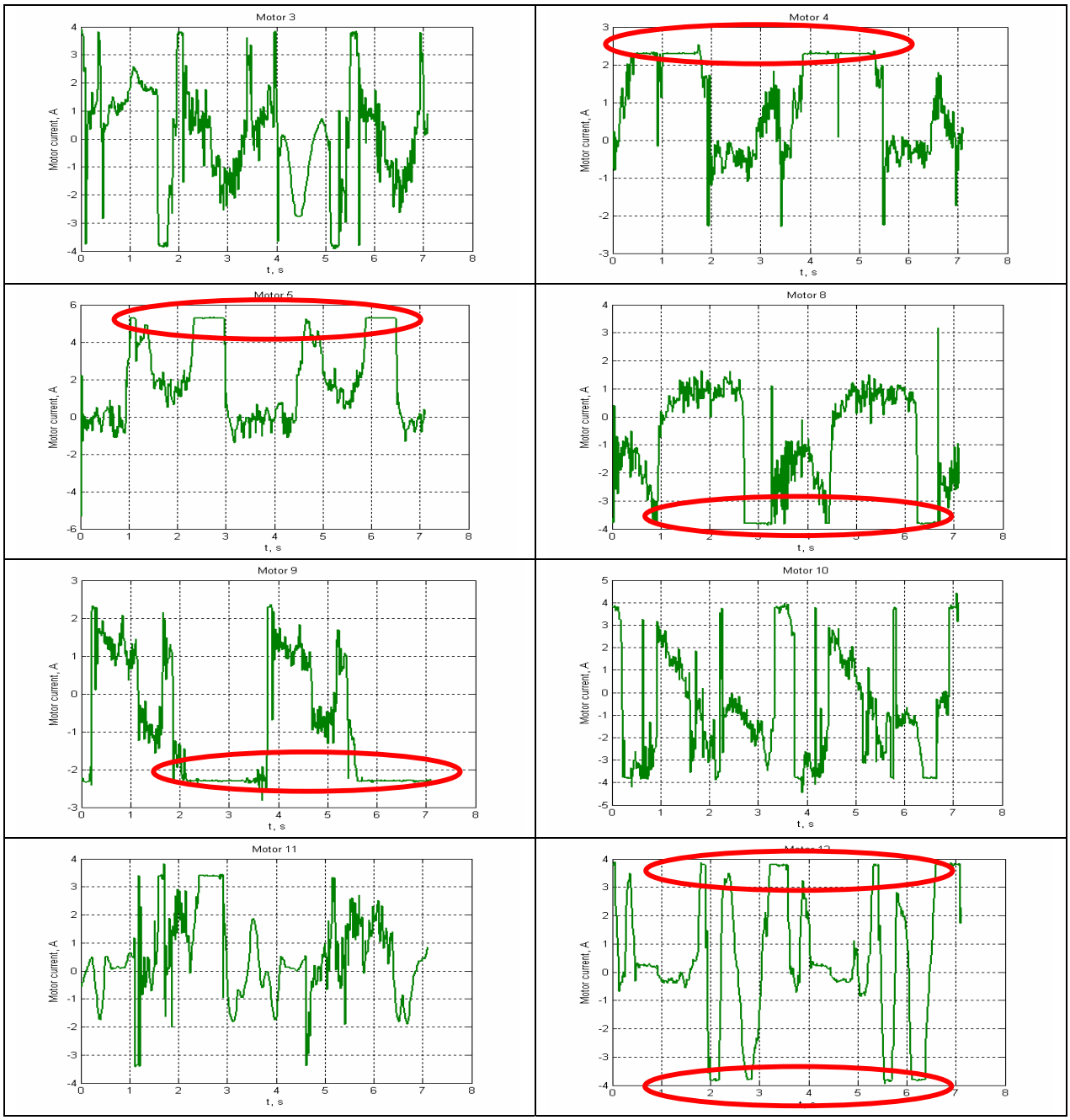


Fig. 7.15: Joint current variation

As can be observed from figure 7.15, the motor current oscillates radically and sometimes approaches its allowed maximum value. In this case, the intelligent motion controller limits the current in order to protect the motor from damage. It can be observed that these limits are achieved in many segments of the path. This means that in some parts of the walking trajectory, the joints need more torque than the motors are able to provide.

Therefore, the worst walking results are caused by the lack of the joints' torques, because of the insufficient power of the installed into the Rh-1 humanoid robot prototype DC motors. It occurred because the additional load (batteries and plastic cover) was added and the dynamic characteristics of the robot changed. An increase in the maximum allowed motor torques (especially at the hip joints) could improve the walking, as well as the implementation of stabilization control algorithms presented above.

7.3 Humanoid robot HRP2

In this work, the physical realization and implementation of the stabilizer was made for the HRP-2 humanoid platform, although it can be easily adapted for most of the existing humanoid robots.

The HRP-2 is the robotic platform for the Humanoid Robotics Project headed by the Manufacturing Science and Technology Centre (MSTC) of Japan. The total robotic system was designed and integrated by Kawada Industries, Inc. together with the Humanoid Research Group of the National Institute of Advanced Industrial Science and Technology (AIST).

7.3.1 Mechanical structure

As it was mentioned in previous chapters, the HRP-2's height is 154 cm and its mass is 58 kg including batteries. It has 30 degrees of freedom (DOF) including two DOF for its hips (Figure 7.16).

The design concepts of the HRP-2 are light and compact, but performable for application tasks like cooperative outdoor projects [Kaneko 2004]. As a result, HRP-2 is designed to be a feminine size. The cantilevered crotch joint allows for walking in a confined area. Its highly compact electrical system packaging allows it to forgo the commonly used "backpack" used on other humanoid robots. The table 7.2 shows the distribution of DOFs inside the robot.

Dimensions	Height	1539 [mm]
	Width	621 [mm]
	Depth	355 [mm]
Weight including batteries		58 [kg]
D.O.F.		Total 30 D.O.F.
	Head	2 D.O.F.
	Arm	2 Arms x 6 D.O.F.
	Hand	2 Hands x 1 D.O.F.
	Waist	2 D.O.F.
	Leg	2 Legs x 6 D.O.F.
Walking speed		Up to 2.5 [km/h]

Table 7.2: Principal specifications of HRP-2 humanoid robot

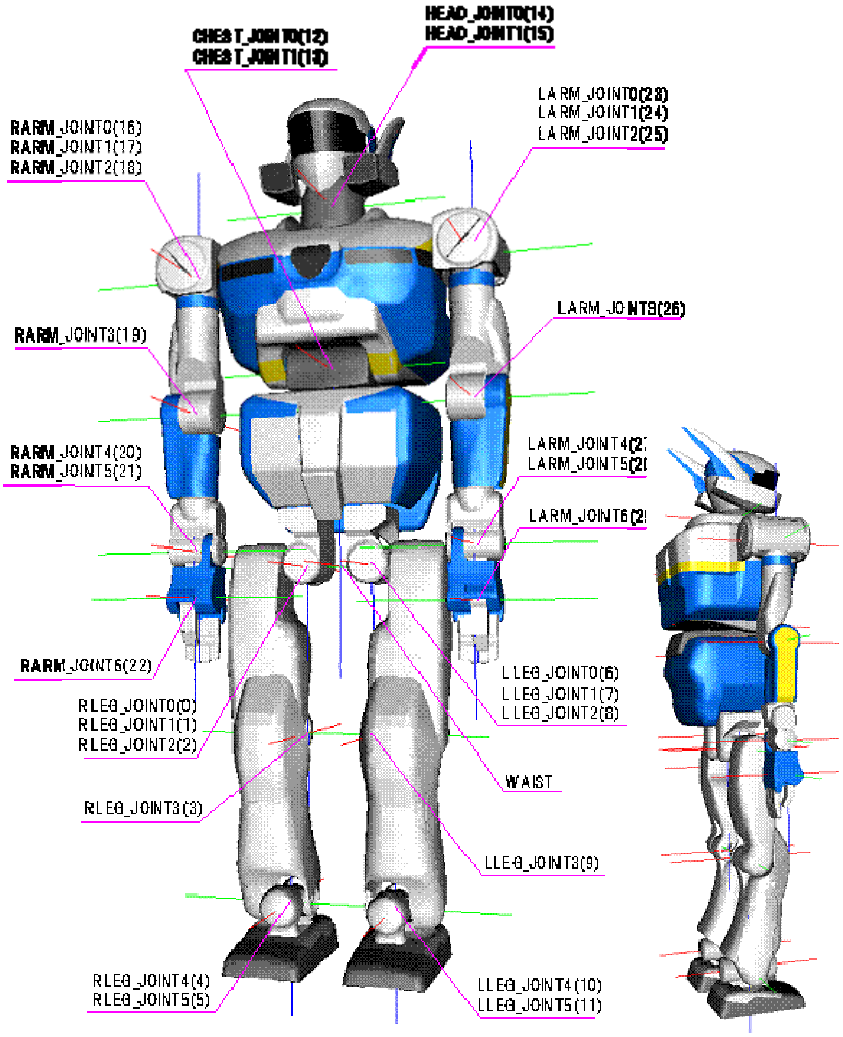


Fig. 7.16: HRP-2 humanoid robot and its joint location

External appearance of the robot was designed by a mechanical animation designer to make a human feel friendly for HRP2. The design policy for the movable range of each joint is designed to be about the same as that of a standard human so that a humanoid robot performs human tasks as well as a human.

7.3.2 Software environment

The main advantage of the HRP-2 humanoid platform is the availability of the software environment OpenHRP. OpenHRP is implemented as a distributed servers system communicating with each other using CORBA (Common Object Request Broker Architecture). CORBA allows servers including dynamics simulators, to communicate with each other using something like a network bus called the ORB. This provides an abstraction layer that allows servers written in different languages or running on different operating systems to communicate and share data (Figure 7.17).

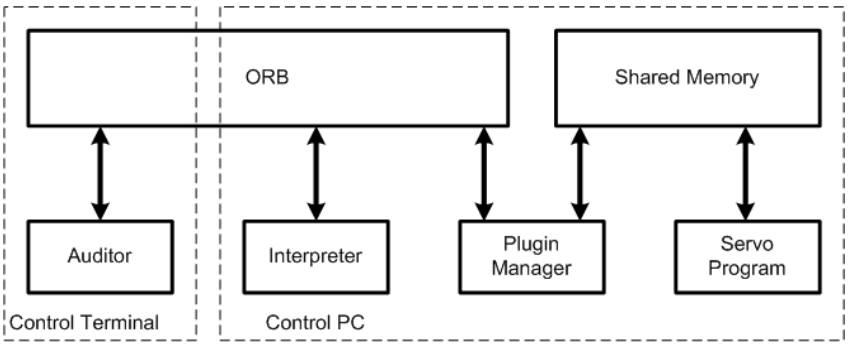


Fig. 7.17: OpenHRP software control system

The Auditor GUI runs on the Control terminal. It monitors the state of the robots various sensors (or simulated sensorial data in the case of the simulation mode) and sends the script file to the Jython Interpreter working on the Control PC mounted into the humanoid robot HRP-2 (in the case of the simulation mode, it can be the same PC as the Control Terminal). The Jython Interpreter receives scripts from the Auditor and executes them.

The plugin manager performs the calculations for controlling the robot in real-time every 5[ms]. It obtains information from the servo level via the shared memory interface and sends the calculated new joint angles to the servo level through the shared memory. The Servo actually performs I/O with the hardware, and operates in real-time at 1[msec]. It receives the A/D values converts them to actual units and sends them and the joint angles to the shared memory. It then receives the joint angles output from the plugins running in the plugin manager and sends them to the I/O board. In the case of the simulation mode, the data is received from and sent to the dynamic simulator directly through the shared memory.

All functionalities of the humanoid robot are implemented as plugins. The calculations are performed in order by calling a specific function in the plugin. Figure 7.18 shows an outline of how the plugin manager works. This system is a kind of centralized control architecture where the plugin manager controls all processes.

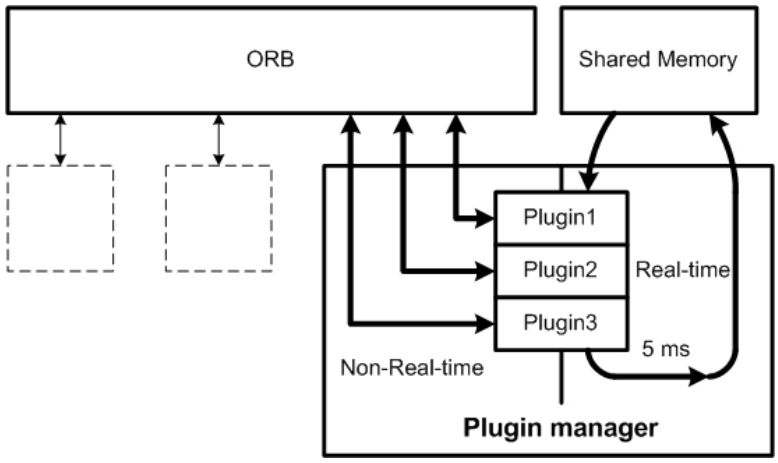


Fig. 7.18: The plugin manager structure

The internal state of each plugin is divided into the real-time part and the non real-time part. The control thread is a real-time thread, which loops every 5[ms]. At the beginning of the loop it first reads information from the sensors and holds this information as the Robot State in a global variable. It then calls the control functions in all the plugin modules in series. The control function has two possible variables: the Robot State, which holds the current sensor values, and the Motor Command, whose control function writes the next goal position for the servo controllers. Every plugin has access to this data and in that way the data modification and exchange between different plugins can be realized. Finally, when the control functions in all plugin modules have been called, the Motor Command is sent to the servo controllers.

The user interface of the OpenHRP system is realized as a program entitled Auditor. This version of the Auditor is implemented in the GUI, which resides in an external control terminal. It uses OpenHRP to conduct simulations and provides an interface to the actual robot. Figure 7.19 shows the screenshot of the Auditor program.

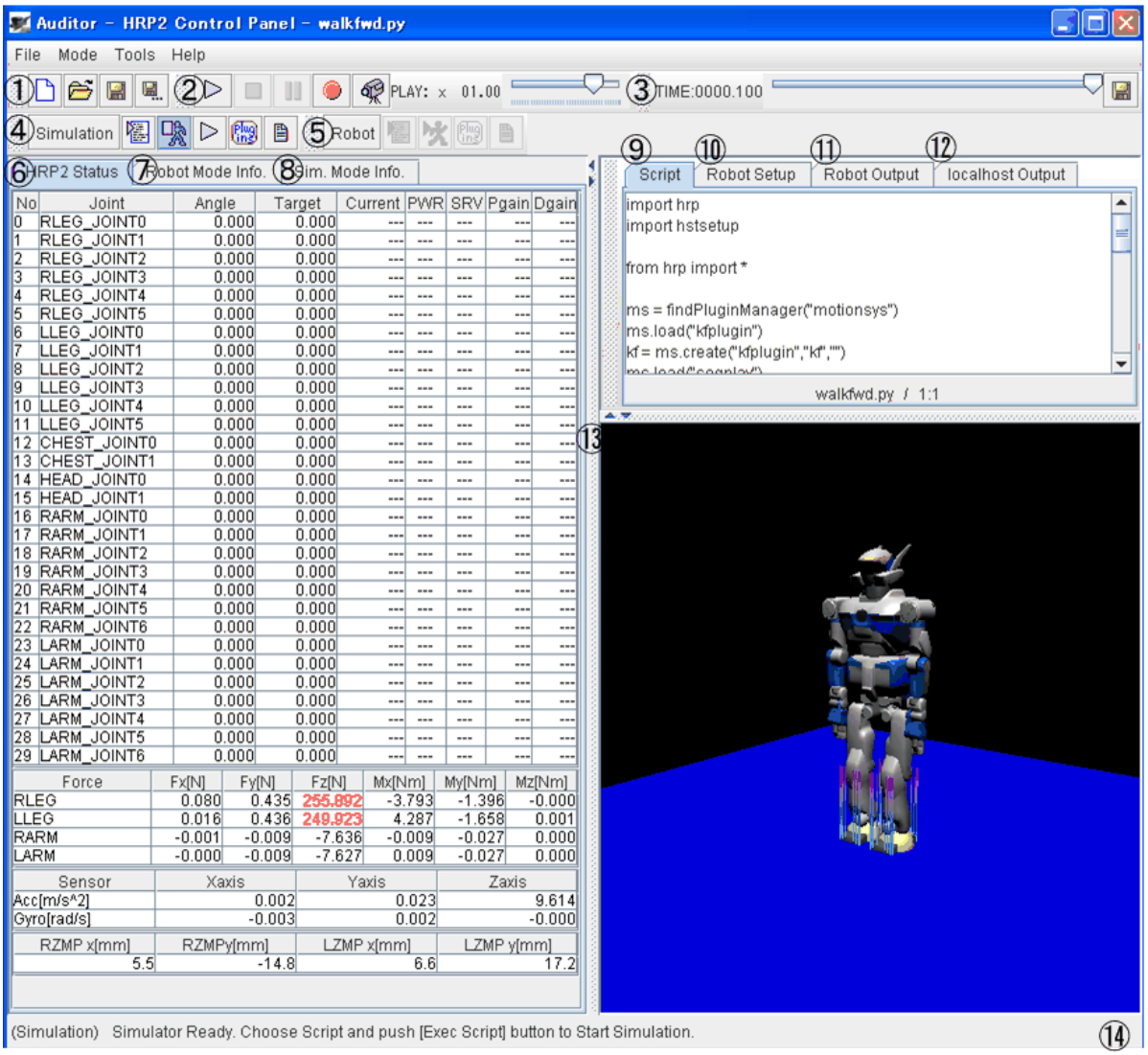


Fig. 7.19: Screenshot of the Auditor

The main tasks provided by the Auditor Interface are the User Terminal and the Motion Control.

User Terminal tasks include:

- Start the OpenHRP CORBA servers
- Load the control software script file
- Send the script to the controller for execution
- Pause simulation according to user commands

Motion Control tasks include:

- Sending script to the real robot
- Monitoring the sensors and joint angles of the robot
- Turning joint servos control on and off
- Calibration of joint angle encoders and sensors

The Auditor is designed so that the user is forced to run everything through a simulation first. Logically, even simple programs can have a bug and this can cause damage to a robot. It may seem time consuming, but this is the safest way to work with a system as complex as a humanoid robot.

7.3.3 Stabilizer implementation

In this Ph. D. thesis, the physical performance and implementation of the Stabilizer is done for the HRP-2 humanoid platform, although it can be easily adapted for most existing humanoid robots. The design and software/hardware architecture of the HRP-2 humanoid robot were explained in the chapter 3.

Figure 7.20 shows the main plugins, which were used in this project, and the data exchange between different modules.

Plugin modules are implemented as C++ classes. They include methods that are called from real-time tasks, and methods that are called from script files. The main plugins, which are needed for the humanoid to walk, are provided by the OpenHRP platform. In this project, the only plugin created and implemented totally was *My_Stabilizer* plugin, which contains the stabilization control algorithm discussed in previous chapters. The rest of the auxiliary plugins used in this project are *hwalk* plugin, *kfplugin* and *ZMPsensor* plugin. The mathematical and computational background for the performance of these modules was developed and explained above, but was not implemented as plugins yet. It should be noted that all plugins provided by the OpenHRP platform were used as black boxes and no information about its internal structure was provided by the manufacturer. This absence of the information on the realization of key functionality of the humanoid robot is the major disadvantage of the OpenHRP system (it encouraged the author to design new open architecture that is the topic of this Ph.D. thesis work).

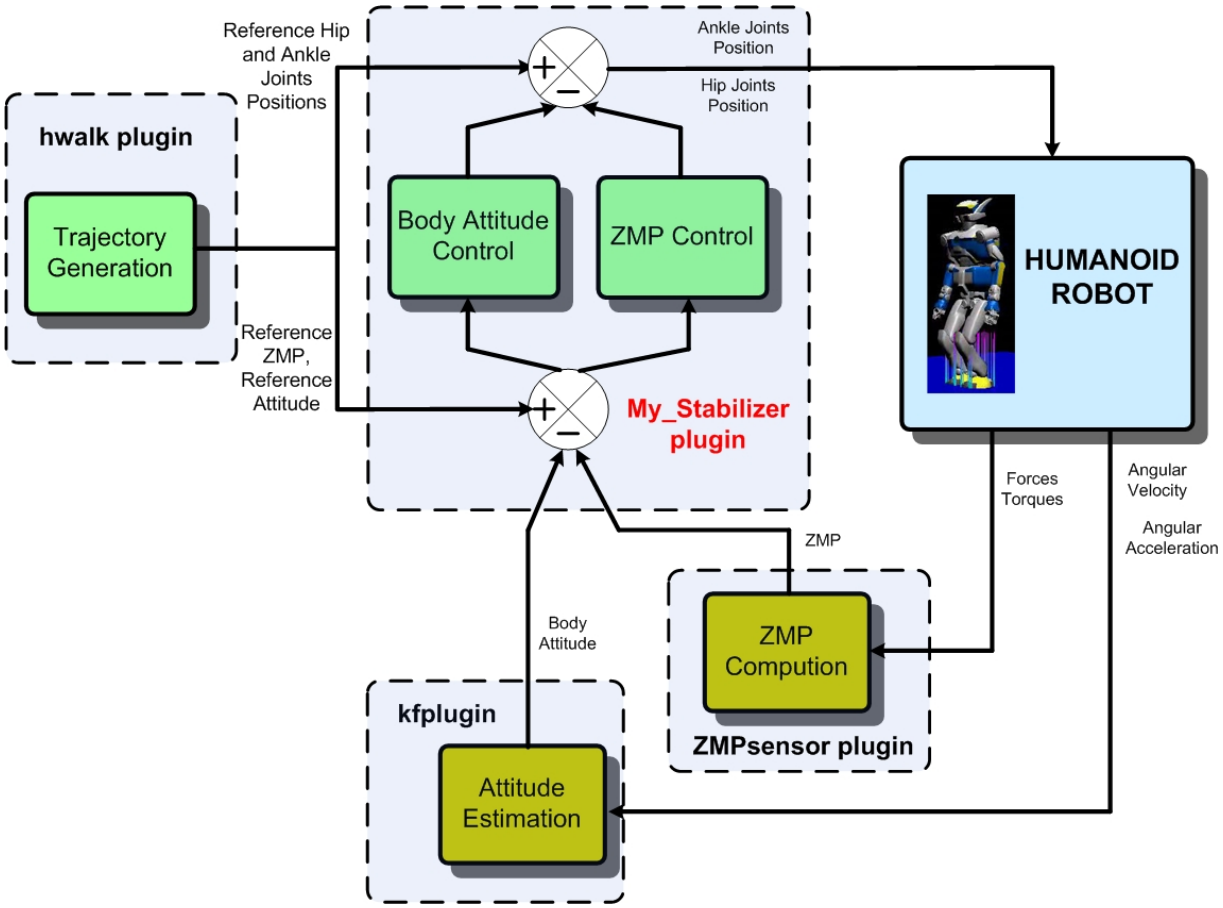


Fig. 7.20: Stabilizer performance as plugins

Hwalk is the standard plugin, which generates online walking patterns – the ideal reference trajectories that are used as input data for the *My_Stabilizer* plugin and also is the input reference for the dynamic model of the robot. The result is saved in the *Motor Command* (*mc*) structure and can be used from all the other plugins.

Kfplugin is the performance of the attitude data processing system. This plugin estimates the robot's real attitude from the current sensors output. If the *kfplugin* is used for simulation it simply uses the values from the Dynamics server. The result is stored in the *Robot State* (*rs*) structure and can be used from all the other plugins.

ZMPsensor plugin is the realization of ZMP computing. It uses output from the force sensors in the feet to calculate the ZMP point in Waist Coordinates. The output from this calculation is written to the Robot State and can be used from all plugins.

In this project, both the *kfplugin* and the *ZMPsensor* plugins were used for providing the actual ZMP value and actual attitude for the *My_Stabilizer* plugin where it was compared with the reference (ideal) state from the *Hwalk* plugin. *My_Stabilizer* performs the computation of errors in actual ZMP and attitude positions and provides its correction by changing the reference positions of ankle and hip joints by the algorithms explained in the chapter 5. All changes are written into the *Motor Command* structure and finally, when the control functions in all plugins

have been terminated, the *Motor Command* is sent to the servo controllers. Thus, with the sample time of 5[ms], the stabilization control task is cyclically performed providing stable bipedal locomotion for the humanoid robot. The following section shows some simulations carried out with the model of the HRP2 humanoid robot provided with the designed stabilizer.

7.3.4 Simulation experiments

All experiments were provided with the OpenHRP simulator and the dynamic model of the HRP2 humanoid robot.

Firstly, before the Stabilizer was implemented, some experiments were carried out in order to find any relationship between the angular motion of hip and ankle joints and Attitude and ZMP positions. The robot was moved into a half sitting position (the initial position for walking which is needed to move the COG into the centre of the contact polygon) and periodical reference input motion for ankle and hip joints was sent simulating inertial and disturbing forces as is shown in figure 7.21.

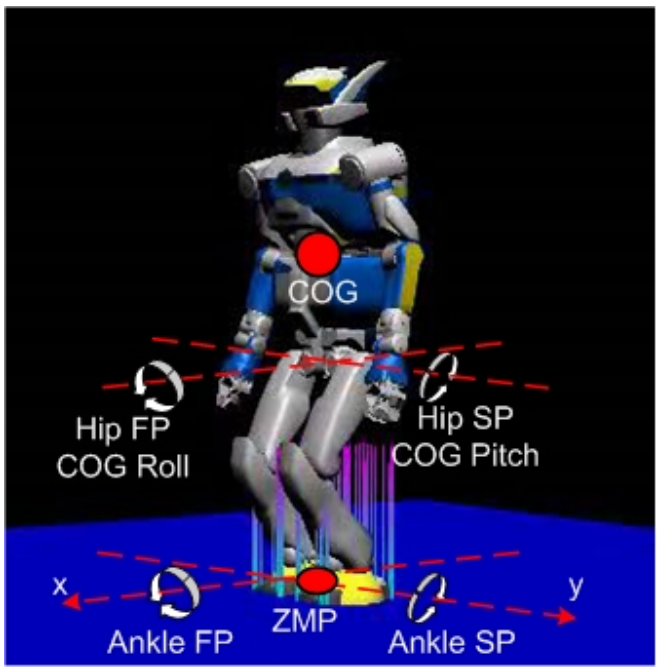


Figure 7.21: HRP2 angular motion

In the case of hip motions, the angular position of the trunk (COG) was measured. The curves in figure 7.22 show how the angular motion of the hip joints in the frontal plane (figure 7.22(a)) and sagittal plane (figure 7.22(b)) changes the attitude of the robot.

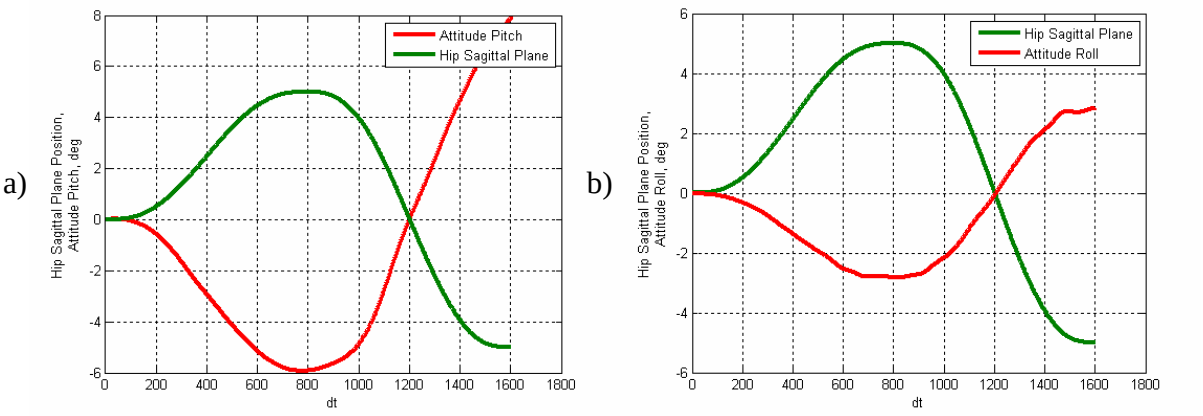


Fig. 7.22: Hip joints and attitude variations a) Hip sagittal plane / Attitude Pitch b) Hip frontal plane / Attitude Roll

There is some dependence between the motion of the hip joint and the attitude of the humanoid robot, which can be observed in the previous figure. In the sagittal plane, the attitude of the trunk takes almost the same angular position as the hip joint of the robot. In the frontal plane, the attitude changes slowly. Further experiments and system identification using Matlab showed that in both cases the dependence tends to be near linear. It allows us to use only the hip in order to correct the attitude. The main advantage is that the hip is located near the actual COG position and its motion has the influence on the upper body directly, without necessity of computing the inverse kinematics for the entire robot.

In the case of ankle motions, the real ZMP was measured. The curves in figure 7.23 show how the angular motion of the ankle joints in the frontal plane (figure 7.23(a)) and sagittal plane (figure 7.23(b)) changes the ZMP of the robot.

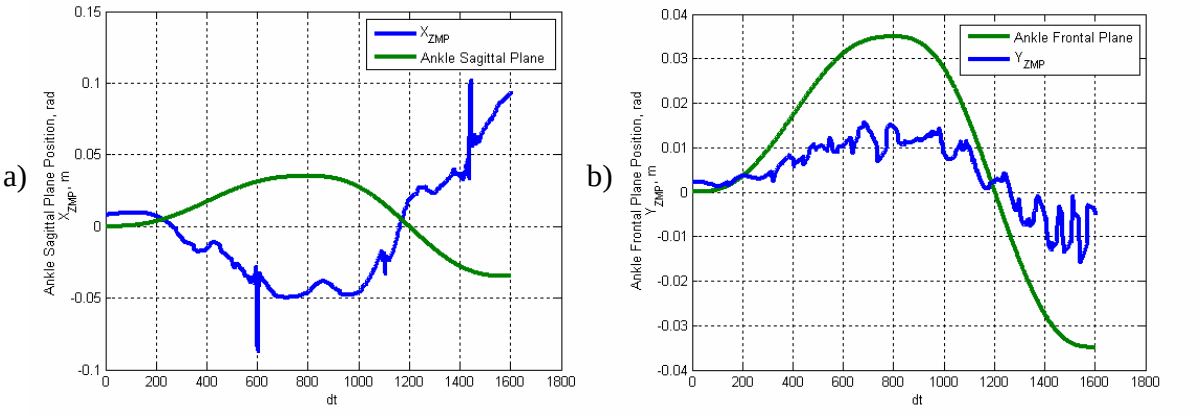


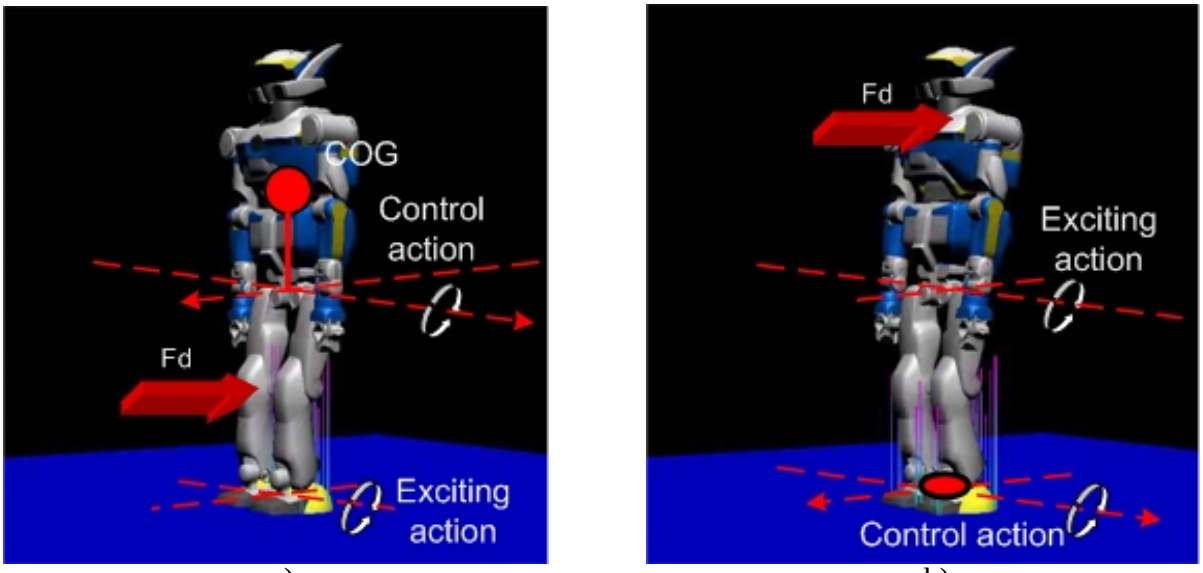
Fig. 7.23: Ankle joints and ZMP variations a) Ankle sagittal plane / x_{ZMP} b) Ankle frontal plane / y_{ZMP}

It can be observed that the smooth motion of the ankle joint provokes rather disturbed ZMP variations with some peaks, which can move the ZMP into the instability zone. Nevertheless,

dependence can also be observed. This dependence allows controlling the ZMP through the motion of the ankle. One of the advantages of this method is that the ZMP is controlled near the contact point of the robot with the ground. Variation of the ankle joint directly shifts the point of implementation of the ground reaction force applied to the foot (ZMP) in the direction needed for the stability of the mechanism.

These experiments showed that the ZMP as a control parameter is more sensitive to external influence and needs very precise tuning of the controller. The attitude of the trunk shows more stable results (action/response) and is easier to stabilize in the desired (vertical) position.

The next series of experiments was carried out with the model of the HRP2 robot provided by the implemented stabilizer. In order to test the functioning of the stabilizer in a static (non-walking) position, the robot was exposed to external disturbances. Both parts of the stabilizer (ZMP and Attitude controls) were tested independently in order to be able to simulate the disturbing force. For the Attitude control, disturbing force *Fd* was applied to the bottom part of the humanoid's body and was simulated as the ankle's motion (Figure 7.24(a)). In the case of the ZMP control, the disturbing force was applied to the upper part of the humanoid's body and was simulated as the hip's angular motion (Figure 7.24(b)). In both cases, the exciting input signal was performed as a periodical movement with a 5-deg amplitude for Attitude control and 3 deg for the ZMP control (as exciting was shown in the previous experiment, ZMP position is very sensitive to any change of the upper body).



a) b)

Fig. 7.24: Static test of the stabilizer in the sagittal plane a) attitude control b) ZMP control

Figures 7.25 and 7.26 present results for the static test of the Attitude and ZMP controls of the dynamic model of the HRP2 humanoid robot.

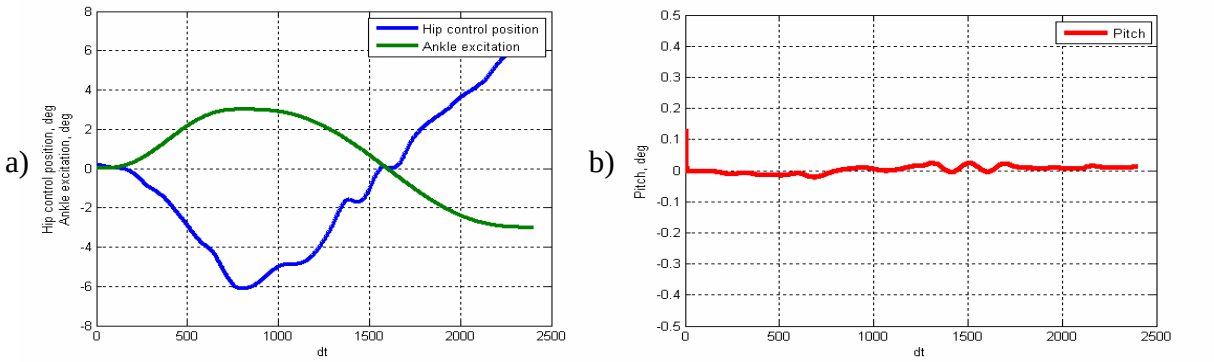


Fig. 7.25: Static test for Attitude control a) Hip control position and Ankle excitation variation b) Pitch variation

From figure 7.25(b) it is clearly seen that the variation of pitch angle of the trunk is minimal (less than 0.02 deg) and it is always in a vertical position for every stage of the ankle motion (or in another words, for every external force applied to the robot). It proves the reliability of the designed Attitude controller in a static case.

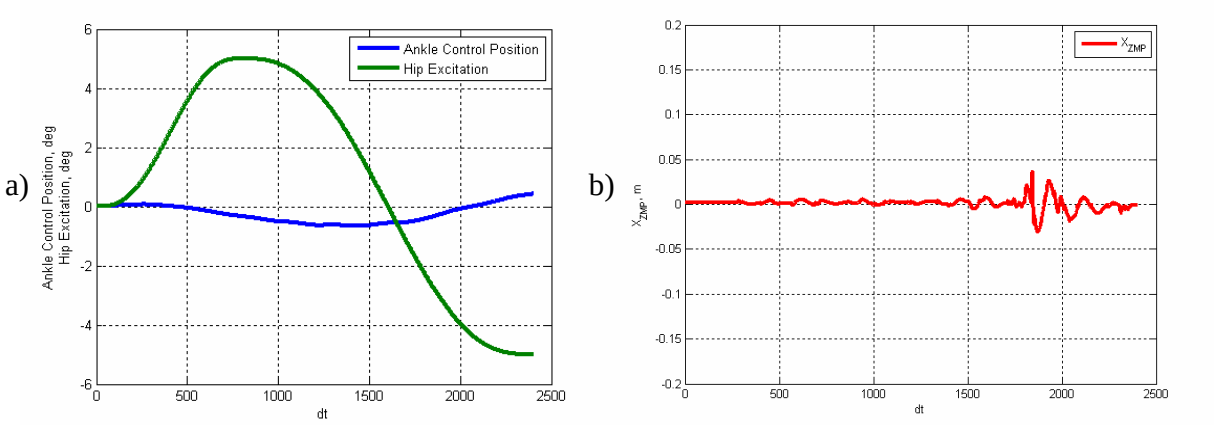


Fig. 7.26: Static test for ZMP control a) Ankle control position and Hip excitation variation b) x_{ZMP} variation

From figure 7.26(b) we can observe that the variation of the ZMP under the Ankle control is rather smooth. In the final phase, some perturbations in the ZMP curve occurred. But after a few moments, its amplitude was attenuated and finally, the ZMP settled near the zero, which is a goal for the ZMP control in the current posture of the humanoid.

Variation of the ankle control position in figure 7.26(a) is smooth; this means that the control action will not bring a lot of modifications in the ankle's motion under the motion pattern in a case of real walking situation. On the other hand, variation of the hip control position in figure 7.25(a) is more notable and will bring more appreciable modifications into the hip position under the motion pattern during walking. Finally, it means that working together, the Attitude control should have more priority or gain than the ZMP control. This proves the theoretical results

obtained in the previous chapter and allows to decouple the entire control task of the stabilization into two independent processes.

After static control experiment was carried out and both controllers approximately tuned, the final stage of this project was accomplished realizing walking simulations with the model of the HRP-2 humanoid robot. At the beginning, the robot was set in walking motion without any stabilizer. Figure 7.27 shows the snapshot of this kind of walking.

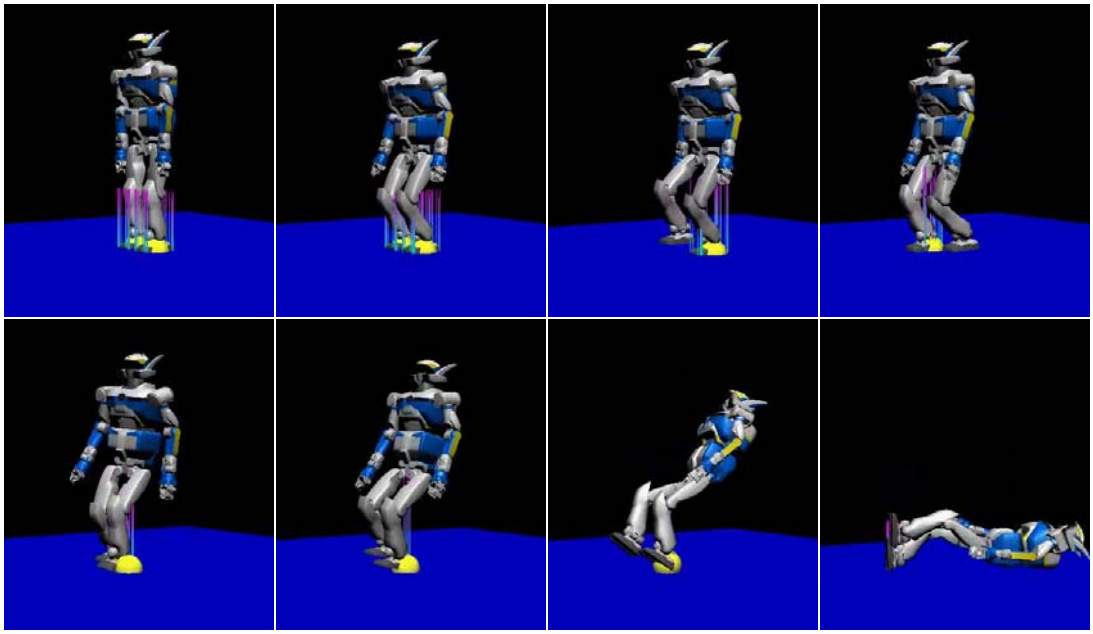


Fig. 7.27: Snapshots of humanoid simulator. Walking without stabilizer

As can be seen in figure 7.27, the humanoid robot starts in a vertical position. In the beginning of its motion it takes a half-sitting posture, which is used for generating stable motion patterns. As this operation doesn't provoke a lot of lateral accelerations, the robot maintains its balance. After this, walking begins. Even the first step evidently shows that the dynamics of walking disturb the balance and the previously generated statically stable walking patterns fail. After the second step, because of the high vibration and cumulative errors in the ZMP position, the robot falls down. This simulation clearly shows the need for the stabilization mechanism, which will correct dynamic errors in the humanoid's walk.

The next figure shows snapshots of the walking simulations of the HRP-2 humanoid robot provided with a developed stabilization system.

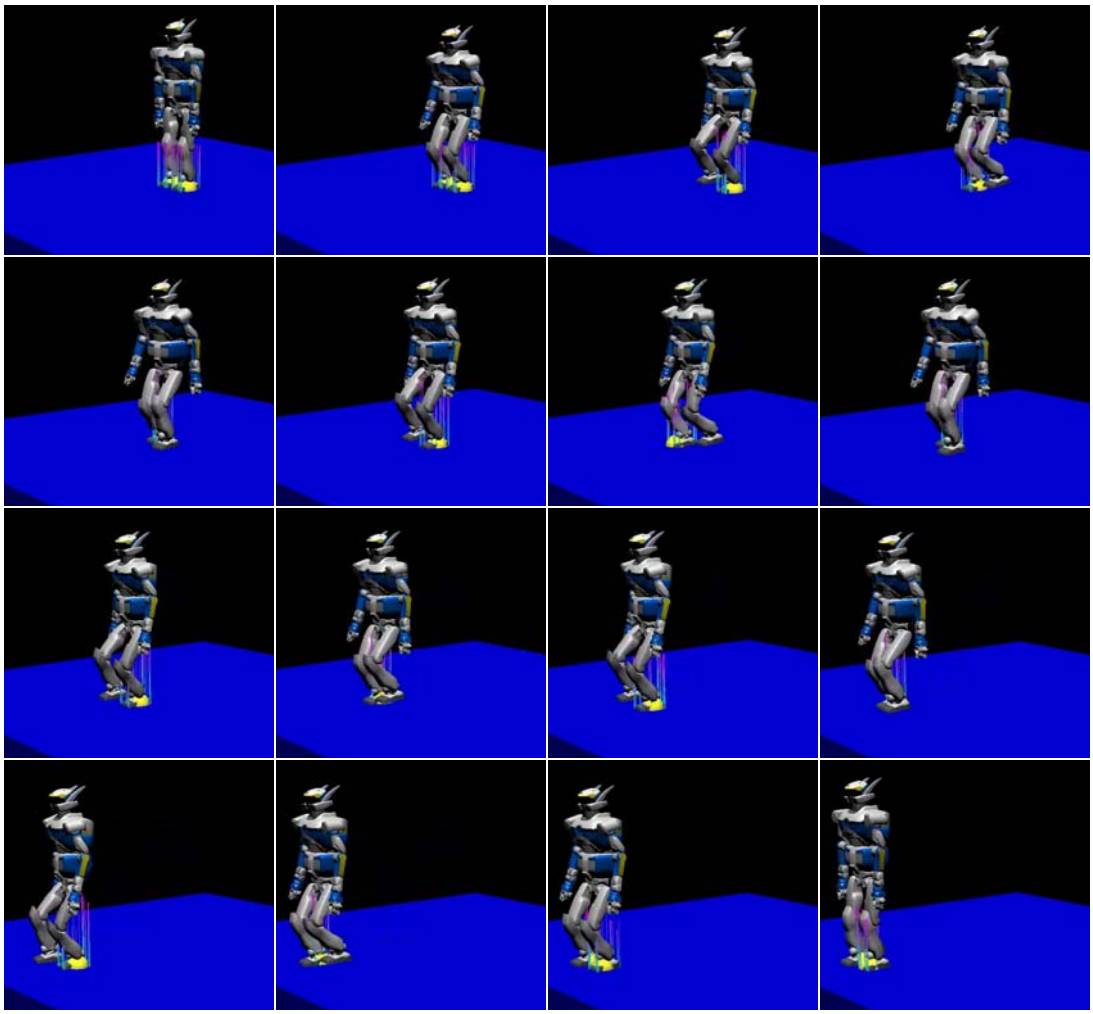


Fig. 7.28: Snapshots of humanoid simulator. Walking with developed stabilizer

As in the previous test, the robot starts from the initial vertical position. At first, it takes the half-sitting position. During this stage a little vibration of the robot can be observed, because the controller tries to adjust trunks and ZMP position for the new COG location. After this, walking starts. The robot takes 10 steps in order to cover a distance of 2 meters. After the goal is reached, it takes an initial vertical position. Finally, figure 7.28 proves that the model of the humanoid robot HRP2 provided with the designed stabilization system can walk stably in a forward walking mode.

Figure 7.29 shows variation of ZMP - principal control parameter of the robot during forward locomotion.

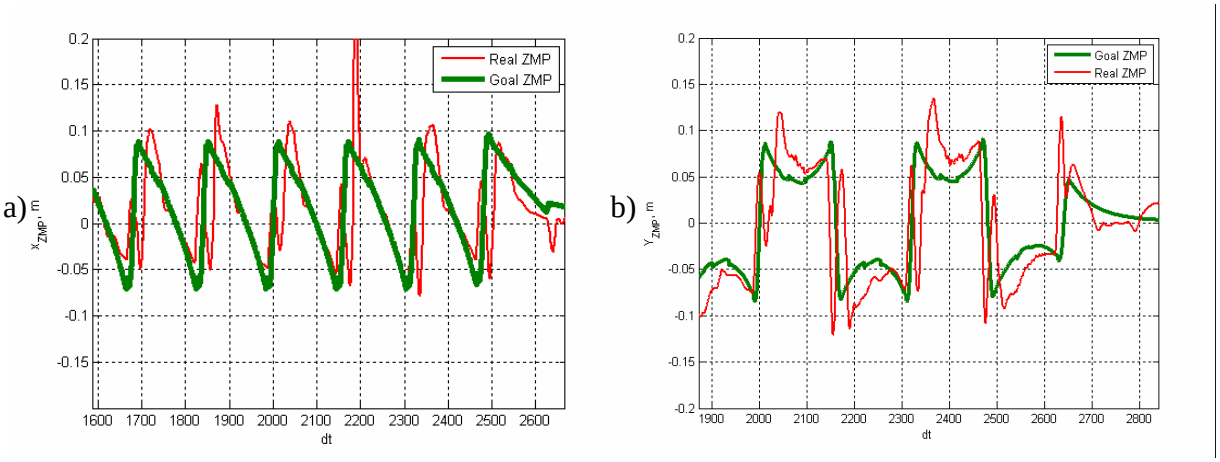


Fig. 7.29: Reference and real ZMP variation a) x_{ZMP} b) y_{ZMP}

In this figure, the reference (previously computed) ZMP is superposed by real (measured by the force-torque sensors) ZMP in both directions of the walking plane. Periodical essence of the ZMP curves corresponds to the periodicity of steps. It can be seen that the real ZMP in both cases oscillates near the reference values. However, some peaks (in the worst of cases, the ZMP even leaves the stability zone) can be observed. Nevertheless, as experiments show the robot doesn't fall down. It can be related with the fact that for high-speed locomotion (Dynamic Gait) the ZMP computed from the torque sensor, at the landing moment because of the impact can abruptly rise. Therefore, the computed actual ZMP can leave the stable support polygon for some instance without overturning the mechanism [Westervelt 2003] because the mechanical system of the robot attenuates this momentary peak of the ground reaction.

Also, the proximity of the real ZMP trajectory to the reference one depends on the tuning of the ZMP controller. As was mentioned above, it was decided to give a low priority to the ZMP control in order to bring the fewest disturbances as possible to the ankle's motion. The next figure presents the variation of the attitude of the humanoid robot.

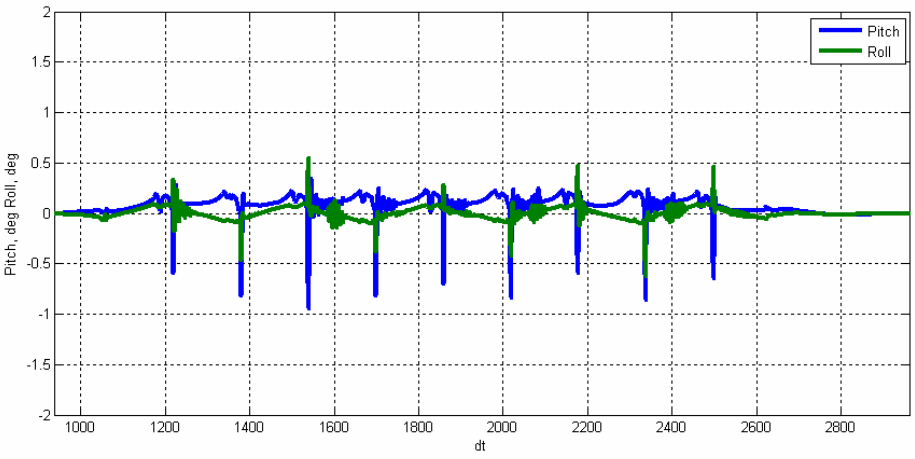


Fig. 7.30: Attitude variation

The Attitude control algorithm supposes that the reference attitude for Pitch and Roll variation is 0. It is clearly seen that the trunk of the humanoid robot is maintained strictly vertical. Peaks in both curves appear as the consequence of the impact of the foot on the ground and even in the worst cases they don't exceed the value of 1°.

Figure 7.31 shows variations of every joint participating in the stabilization control.

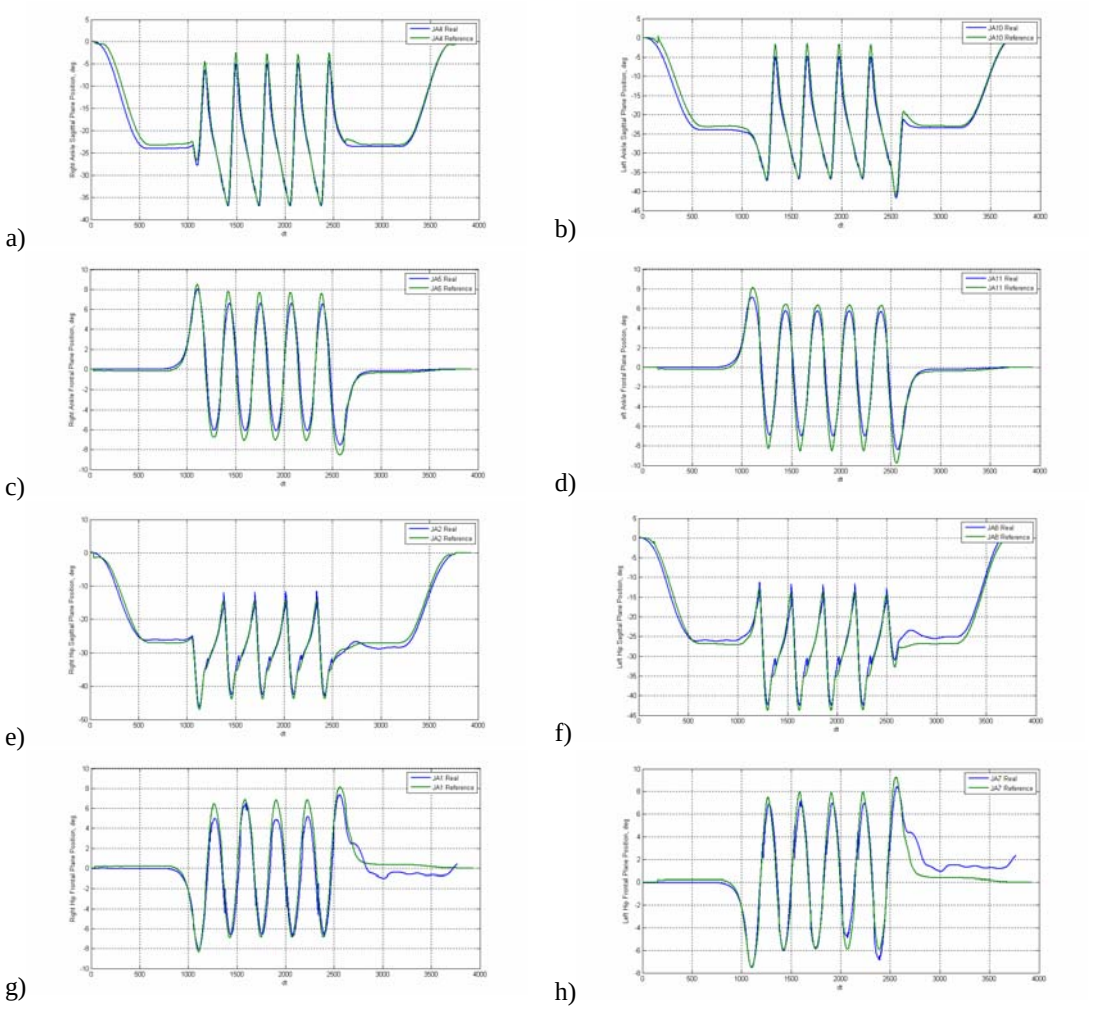


Fig. 7.31: Real and reference joints variations. a) Right ankle sagittal plane b) Left ankle sagittal plane c) Right ankle frontal plane d) Left ankle frontal plane e) Right hip sagittal plane f) Left hip sagittal plane g) Right hip frontal plane h) Left hip frontal plane

From the previous figure it can be seen that the control brings minimal correction into the ideal walking patterns of the hip and ankle joints. Nevertheless, these changes are more pronounced for the hip joints, especially in the final stage of the trajectory when the cumulative error is high and the controller tries to adapt the body to the final goal position. Moreover, the deviation between the ideal and real trajectories is higher in the frontal plane, because disturbing lateral acceleration in this plane tends to be higher and the stabilizer needs to interfere harder.

Further experiments with the model of the HRP-2 humanoid robot provided with the developed stabilization system showed rather good results not only in forward walking, but also in turning and backwards walking gaits. However, it should be noted that in a backward gait the stabilizer showed worse performance and in some experiments the model overturned and felt down. It may be related to changes in the dynamics of the system (backward walking mechanism of the humanoid differs from the forward locomotion) thus, the controller tuned for the forward motion doesn't comply with new conditions. Nevertheless, the experiments performed using the OpenHRP simulator showed good efficiency and reliability of the proposed stabilization control algorithm.

7.9 Conclusions

This chapter presented the implementation of developed motion control hardware and software architecture and control algorithms with the Rh-1 humanoid robot. Rh-1 is a humanoid robot in the second phase of the Rh project. The basic design consideration for mechanical system of the robot was the anthropomorphic body with human-like capabilities in arms movement and biped locomotion. Finally, the mechanical structure of Rh-1 robot consists of two arms, two legs, the trunk and the head, and possesses 21 degrees of freedom.

The open motion control architecture for the Rh-1 humanoid robot was designed as a distributed control system. The hardware system was implemented using the conventional electronic components of the automation industry in order to reduce the development time and cost, and to have a flexible and easily upgradeable hardware system. The distribution of electronics inside the robot's body was made in order to have easy access to every component for cabling, service and maintenance.

According to the software architecture proposed in the previous chapter, the software system was implemented as three basic software modules: Control Server Control, Agent and number of HRSC Clients. All software code for Rh-1 humanoid robot motion control system was implemented using C-based programming languages. The developed software system provides the effective user interface and motion control algorithms allowing the robot to make motions.

In order to prove the applicability of the discussed architecture, different walking experiments were carried out. Performed experiments tested different gaits of humanoid robot motion in the open-loop mode. Experiments showed the stable locomotion of the robot in the dynamic forward walking gait, which is the fundamental type of motion for achieving walking stability. It was proved experimentally that Rh-1 robot provided with the current mechanical, hardware and software control architectures could walk stably. Nevertheless, further experiments with the Rh-1 robot provided with batteries and cover increased the weight of the system and worsened biped locomotion. Improvements in the mechanical structure and implementation the stabilization control algorithm will improve the overall walking characteristics of the Rh-1 humanoid robot.

The physical performance and implementation of the stabilizer was made for the HRP-2 humanoid robot with OpenHRP software platform. The main advantage of the OpenHRP is that it can be implemented for simulations not only with HRP-2, but also with every humanoid platform. Therefore, the research work (not only the design of the stabilizer, but also the plugin development and programming) done during this Ph.D. thesis work can be applied in the future for the development of other humanoid robots.

The simulations carried out with the model of the HRP-2 provided with developed stabilization system showed good results not only in forward walking but also in turning and backwards walking gaits. It proved the applicability and reliability of the designed stabilizer. Further work will be centred on the implementation of a developed stabilizer with the Rh-1 humanoid robot in order to provide it with more stable dynamic locomotion.

Chapter 8

Conclusions and future work

CHAPTER 8

Conclusions and future work

8.1 Conclusions

As mentioned in the introduction, this thesis has focused on the study of motion control architecture for humanoid robots. Because of the considerable future benefits, research on humanoid robots is a very hot topic in engineering and science society today. The development of humanoid robots still remains a very complex engineering and scientific task which requires an implementation of new approaches in different fields, such as mechanical design, electronics, software engineering and control. Nowadays, humanoids have a great impact on many areas of engineering, across many disciplines and fields of study in robotics. Moreover, the rapid development of humanoid intelligence and capability raises some serious ethical questions which are a topic for many philosophical and technical studies.

Some main conclusions derived from the work carried out here can be grouped into the following points:

- First of all, this thesis provides a complex study of a control architecture design and a proposal for generalized open motion control architecture for humanoid robots. This autonomous motion control architecture is based on the modular distributed hierarchical structure. The benefit of this architecture is the possibility of easy and effective control of a hyper DOF system such as that of a humanoid robot. This approach makes the architecture very versatile and adaptable for future modifications and improvements.
- It has been shown that motion for a humanoid robot means biped locomotion; moreover, the thesis proposes the requirements for the design of motion control architecture of a humanoid robot. Since a humanoid robot is a multi joint system, it has a joint control problem for a multi-axis system. The motion of every joint changes the dynamics of a robot, thus, influencing other joints. Furthermore, the implementation of only servo control for some types of robots (especially for walking systems) is not sufficient. Even having stable motion patterns and well-tuned joint control, a humanoid robot can fall over while walking. Therefore, these robots need the implementation of another, upper control loop (Stabilizer) to stabilize their motion.
- The joint control problem of a humanoid robot can be solved by different modes. In this study a dynamical model of the system and classical control theory has been used. The thesis provides a study on identification of a humanoid robot joints and considers as well the possibility of decoupling the dynamics of joints of the robot. Using the influence of the reduction ratio to the dynamical model, it has been proved that the decoupled joint

dynamics can be used without any loss of the quality of control. The main objective of the design is to obtain fast and precise position responses for joint controllers. Practical aspects of the joint controller design have also been considered. Implemented cascade control scheme provides acceptable control characteristics for the system.

- Further in-depth study of the influence of the load torque and the moment of inertia of the link on the system's behavior has been carried out. Experiments show that because of the continuous change of the joint's load and moment of inertia during walking, the system's response vary, although not appreciably. Thus, the adaptive gain-scheduling controller was designed. In same working conditions it shows better control performance than the standard static parameters controller, but taking into account its high implementation and computational costs, the non-necessity of joint adaptive control for the humanoid robot biped walking was confirmed.
- Although planned motion patterns satisfy the stability constraints, certain errors caused by the dynamics, irregularity of the terrain or external forces can cause the humanoid robot to fall down. The most important criterion for determining the stable locomotion of a humanoid robot is the ZMP concept. One of the possible ways to generate stable motion patterns is the 3D-LIPM model which uses the dynamics of the inverted pendulum in order to model bipedal walking. Further studies of the humanoid's locomotion have shown that the most effective way to control the stability of a humanoid robot is the following: provide ZMP correction of the trajectory by controlling the motion of the supporting ankle joint and at the same time control the COG position maintaining the attitude of the Upper body through correction of the hip's trajectories. This control can be modeled as a double inverted pendulum. A mathematical model of the stabilization control of a humanoid robot has been developed and the controller designed. Furthermore, a novel simple decoupled approach has been proposed. This approach considers that ZMP and Attitude parts can be implemented as totally independent and their influence on each other can be compensated by the controller. A variety of sensors used for ZMP and Attitude measurements have been discussed and the design of the data processing system proposed.
- The motion control software and hardware architecture for humanoid robots has been proposed; this was designed using an open systems approach. The basic advantage of open architectures is that it promises a faster and more economical development of high-quality humanoid robotics platforms which are technologically up-to-date. The architecture developed here fulfils the requirements for open, hierarchical distributed architectures. Altogether, the proposed system, including hardware, software systems and communication infrastructure, allows a humanoid robot to produce synchronized multi-axis dynamical walking, performing the real-time joint and stabilization controls.
- Developed motion control hardware and software architecture and control algorithms have been implemented with Rh-1 humanoid robot. The hardware system was implemented using the high quality commercial electronic components of the automation industry in order to reduce the development time and cost and to have a flexible and easily upgradeable hardware system. According to the proposed software architecture, the system was implemented as three basic software modules: Control Server, Control Agent, and number of HRSC Clients. In order to prove the applicability of the discussed

architecture, different walking experiments were carried out. It has been proved experimentally that the Rh-1 robot equipped with the current mechanical, hardware and software control architectures can walk stably. The physical realization and implementation of the stabilizer was also made for the HRP-2 humanoid robot with OpenHRP software platform. The simulations carried out using the model of the HRP-2 provided with a developed stabilization system showed good results not only in forward walking but also in turning and backwards walking gaits, thus proving the applicability and reliability of the designed stabilizer.

As a conclusion let us highlight the main contributions of the thesis:

- Novel open motion control architecture for humanoid robots has been developed.
- New efficient algorithms for motion control and stabilization of the humanoid robot walk have been proposed. Experiments provided with different humanoid platforms have shown that the proposed control architecture can be used to improve the robot's walking.
- The proposed open software and hardware architecture has been successfully implemented with the new humanoid platform Rh-1.
- The architecture developed here offers a universal open solution and could be easily adopted for implementation in other humanoid projects.

Finally, it should be mentioned that the author believes that the major contribution of this thesis is a complex profound study which considers the motion control system of a humanoid robot as a whole. It stresses a whole concept-design-implementation chain, structures the related studies and tries to propose original and efficient solutions, which allows this work to be considered as a methodology or guideline for future developments in the field of humanoid robotics.

8.2 Future work

There are some interesting lines of research opened in this thesis which can be followed up in future work.

It was found that the electromechanical system of a humanoid robot is a key component for achieving a stable bipedal walking. As shown by the experiments provided, the humanoid robot Rh-1 needs a rebuilding of the mechanical structure and the increasing of motor power, especially in the hip joints. This imposes the conditions for the building of the next prototype of the project – the Rh-2 humanoid robot. Solving mechanical and motorization problems will allow the implementation of new motion patterns and produce stable biped walking. Also, further work will be centered on the implementation of developed stabilization control with the Rh-2 humanoid robot in order to provide it with stable dynamical locomotion.

Experiments have shown that the sensory feedback plays a large role in the motion control of the humanoid robot. Future work supposes the implementation of another faster communication system in order to provide the control with more complex real-time information on the current humanoid's state and thus, improve the general stability of the system.

As shown in the previous chapter, the motion control system of most relevant contemporary humanoid robots (HRP-2 as well) is made like a "black-box". This work tries to show the

advantage of open control architecture which provides a universal solution that can be adopted in future humanoid projects. Therefore, Rh-2 humanoid robot will be designed following the same open principles for the motion control architecture as the Rh-1 prototype.

The Hardware system should be rebuilt using more powerful and up-to-date hardware devices. A complete updating of the onboard control system is needed to increase the computing and communicational potential of the system for implementing new advanced motion control strategies. An improvement in the Software architecture should include the further distribution of the robot's control modules by implementing the CORBA interfaces.

Moreover, this study may be improved on what is known about the Stabilizer. There is still much work to do in this area. This Ph.D. thesis considered and developed the Double Inverted Pendulum based model for stabilization control of a humanoid robot. Further research should be centered on improving the proposed control algorithm through the implementation of intelligent control approaches. Moreover, new dynamical models for increasing the efficiency and stability of control algorithms should be studied.

After the model of the systems is improved, other design requirements could be studied. For example, the next research could include the study of new mechanisms for walking on inclined and rough terrains as well as walking control for terrains with changing friction.

Appendix A

PC/104 standard

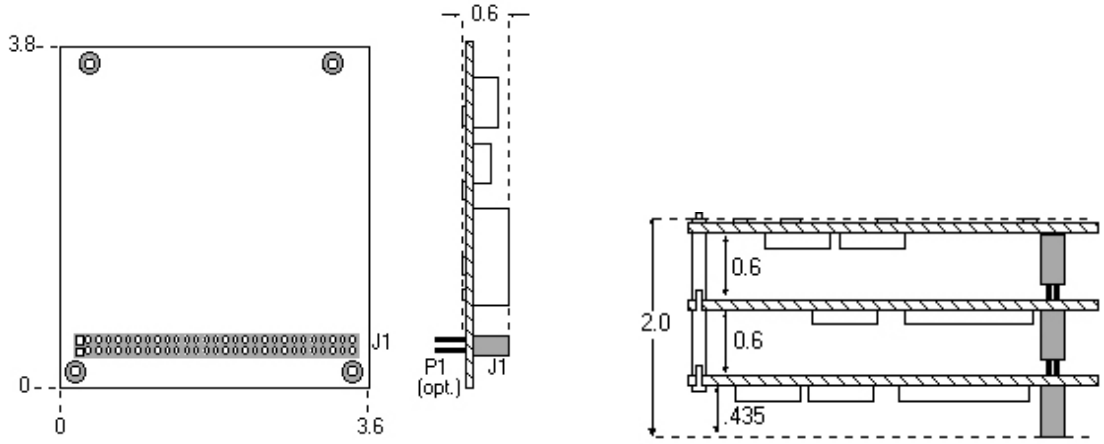
APPENDIX A

PC/104 STANDARD

A.1 Form factor

The PC/104 is an embedded computer standard which defines both a form factor and computer bus. The PC/104 form factor was standardized by the PC/104 Consortium in 1992. An IEEE standard corresponding to PC/104 was drafted as IEEE P996.1, but is not ratified yet.

Unlike the popular ATX form factor which utilizes the PCI bus and is currently used for most PCs, the PC/104 form factor has no backplane, and instead allows modules to stack together like building blocks (figure A.1(b)). The stacking of buses is naturally more rugged than typical PCs. This is a result of mounting-holes in the corner of each module which allow the boards to be fastened to each other with standoffs. The standard size of boards complying to the form factor is 3.55 × 3.775 inches (90.17 × 95.89 mm), while the height is typically constrained to the boundaries of the connectors (figure A.1(a)). A constrained height region guarantees that modules will not interfere with their neighbours.



a) b)

Fig. A.1: PC/104 a) form factor b) stack

Contemporary PC/104 form factor computers with the bus type implemented insight, can be divided into two groups:

PC/104. The PC/104 computer bus (first released in 1992) utilizes 104 pins. These pins include all the normal lines used in the ISA bus, with additional ground pins added to ensure bus integrity. Signal timing and voltage levels are identical to the ISA bus (16-bit), with lower current requirements.

PC/104-Plus. The PC/104-*Plus* form factor adds support for the PCI bus (32-bit), in addition to the ISA bus of the PC/104 standard. The name is derived from its origin: a PC/104-*Plus* module has a PC/104 connector (ISA) *plus* the PCI connector.

From these definitions it is clear that for achieving more flexibility of the control architecture, the PC/104-Plus form factor is preferable due to having several data transmission buses that allow the use of more extensions and peripherals.

A.2 Extension controllers

One of advantages of a PC/104 form factor computer is that while a typical system (also referred to as a stack) includes a motherboard with a mounted x86 based CPU, analogue-to-digital converter, and digital I/O (data acquisition) module, other peripherals can also be found on the market including GPS receivers, IEEE 802.11 controllers and industrial bus controllers. It enables organisation of a complex system into a Control level of the motion control architecture for humanoid robots.

The PC/104 modules constitute almost everything which could be considered necessary in a controller. Therefore, a stack of PC/104 modules can be considered as different controllers executing its tasks and communicating via PC/104 bus, thus, organizing the Control level of the motion control architecture.

Appendix B

Communication technologies

APPENDIX B

Communication technologies

B.1 Communication networks comparison

Tables B.1 and B.2 show the general information, physical characteristics, and transport mechanism of specific communication standards covered in this Ph.D. thesis. These tables were produced using published vendor documents and information from various vendor websites.

Table B.1: Physical characteristics of communication standards

Communication standard	Network Topology	Physical Media	Max. Devices (nodes)	Max. Distance
AS-I	Bus, ring, tree star	Two wire cable	31 slaves	100 meters, 300 with repeater
Interbus	Bus,Tree, Ring, Star, 16 level nesting	Twisted-pair, fibre, slip-ring, Infra-red	512 System Total 256 on remote bus	12.8 Km (copper) total 400m max. between devices
Seriplex	Tree, loop, ring, multi-drop, star	4-wire shielded cable	500+ devices	500+ ft.
CanOpen	Trunkline/Dropline	Twisted Pair +optional signal & Power	127 nodes	1000m
DeviceNet	Trunkline/dropline with branching	Twisted-pair for signal & power	64 nodes	500m
Firewire	No-loops Tree	6-wire shielded twisted pair copper -	capability to address 1023 networks of 63 nodes, each with 281 terabytes of memory	total maximum end-to-end distance of 72m

Macro	Ring based on FDDI (Fibre Direct Digital Interface) networking technology	Fibre optic, coax, or RJ-45	256 -- Supports up to 16 masters, with up to 16 nodes interfaced per master	Glass -- 3000 m Copper -- 30 m
SERCOS	Ring	Fibre optic	254 per ring, multiple rings allowed	Plastic - 40 m node-to-node; 10,000+ m max. ring length. Glass - 800 m node-to-node; 200,000+ m max. ring length.
SDS	Trunkline/Dropline	Twisted-pair for signal & power	64 nodes, 126 addresses	500m
Fieldbus Foundation H1	Multidrop with bus powered devices	Twisted-pair	240/segment, 65,000 segments	1900m @ 31.25K 500m @ 2.5M
Profibus DP/PA	Line, star & ring	Twisted-pair or fibre	127 nodes	24 Km (fibre)
Profibus FMS	Line, star & ring	Twisted-pair or fibre	up to 126 nodes are available and all can be masters if desired	24 Km (fibre)
Profinet	Line, star & ring	Twisted-pair or fibre	127 nodes	24 Km (fibre)
ControNet	Linear, Tree, Star, or combination thereof	Coax, fibre	99 nodes	1000m (coax) 2 nodes 250m with 48 nodes 3km fibre; 30 km fibre with repeaters
Foundation Fieldbus HSE	Star	Twisted-pair, fibre	IP addressing - essentially unlimited	100m @ 100Mbaud twisted-pair 2000m @ 100Mbaud fibre full duplex
Ethernet	Star, Bus	Twisted-pair, fibre optic	1024	100m @ 100Mbaud twisted-pair 2000m @ 100Mbaud fibre full duplex

Table B.2: Transport mechanisms of communication standards

	Communication Method	**Transmission Speed**	**Data Transfer Size**	**Arbitration Method**
AS-I	Master/slave with cyclic polling	Data and power, EMI resistant	31 slaves with 4 in and 4 out	Master/slave with cyclic polling
Interbus	Master/slave with total frame transfer Transparent data linking via master	500kBits/s, full duplex	Up to: 64bytes real time data /device, 244 Bytes PCP data/device 512 In + 512 Out bytes per system	None
Seriplex	Master/slave peer to peer	200 Mbps	7680/transfer	Sonal multiplexing
CANOpen	Master/slave, multi-master, broadcast	1 Mbit	8 byte variable message	bit-wise arbitration of the identifiers
DeviceNet	Master/slave, multi-master, others	500 kbps, 250 kbps, 125 kbps	8-byte variable message	Carrier-Sonac Multiple Access
Firewire	Multi-master, Peer-to-peer, guaranteed and deterministic bandwidth	100, 200 and 400 Mbps with 1394a-2000, and up to 3200Mbps with P1394b	Minimum – 20 bytes header, "payload" extra	request-grant handshake process between a node and every node in the path up to the root node to win a turn at using the bus
Macro	Multi-master - up to 16 per ring	120 MBits/sec	Variable size	Shift register between adjacent nodes
SERCOS	Master/slaves	2/4/8/16 MBits/sec	Variable size, Minimum Master Sync Telegram - 6 bytes	Deterministic time slice, depends on the number of nodes on the ring.
SDS	Master/slave, peer to peer, multi-cast, multi-master	1Mbps, 500 kbps, 250 kbps, 125 kbps	8-byte variable message	Carrier-Sonac Multiple Access

Profibus DP/PA	Master/slave peer to peer	DP up to 12 Mbps (it can also be operated at speeds as low as 9600 baud) PA 31.25 kbps	244 bytes	Token passing
Profibus FMS	peer to peer messaging format	Same as Profibus/DP	More overhead	Token passing
ControlNet	Producer/Consumer Device Object Model	5 Mbps	0-510 bytes variable	CTDMA Time Slice Multiple Access Modified, Device
Fieldbus Foundation HSE	Client/server publisher/subscriber, Event notification	31.25kbps 1 Mbps 2.5Mbps	16.6M objects/device	Deterministic centralized scheduler, multiple backup
Ethernet	Peer-to-peer, Broadcast	10Mbps, 100Mbps, 1000Mbps	Min: 128 bytes Max: 1582 bytes	CSMA/CD

B.2 CAN bus

Controller Area Network (CAN or CAN-bus) is a computer network protocol and bus standard designed to allow microcontrollers and devices to communicate with each other, with or without a host computer. It was designed specifically for automotive applications but is now also used in other areas.

Evaluating the required bandwidth for servo control in a humanoid robot it is obvious that it can be dealt with by one or more buses. CAN bus was chosen because of its characteristics, principally that it can transmit messages in real-time, in other words the message with the highest priority will be delivered within a guaranteed latency time. Other important properties of CAN are:

- Bandwidth up to 1 MBit/s: in a humanoid robot this is sufficient to control 10 to 15 axes (e.g. all feet or all arms of the robot).
- Differential data transmission: this is important to reduce effects of EMI caused by the electric motors.
- Broadcast communication: All messages transmitted are received in all nodes. All receiving nodes decide if they want to accept this message. This guarantees data consistency as all nodes in the system use the same information.
- Large number of nodes.
- Multi-master capability: Any CAN node may send a message, if the bus is idle.

- Sophisticated error detecting mechanisms and re-transmission of faulty messages: This guarantees network-wide data consistency.
- Physical length of connection: at 1 MBit/s cabling can be up to 40 m long. This length will not be exceeded in a humanoid robot.
- Availability: CAN bus is approved in the automotive industry, so highly reliable components are available.

Thus, all these properties make the CAN bus the preferable choice for organizing communication at sensory and device levels of the motion control system for humanoid robots.

The CAN protocol defines the data link layer and part of the physical layer in the OSI model, which itself consists of seven layers. The International Standards Organization (ISO) defined a standard, which incorporates the CAN specifications as well as a part of the physical layer: the physical signalling which comprises bit encoding and decoding as well as bit-timing and synchronization.

The CAN bus uses Non Return to Zero (NRZ) bit coding. In this method the signal level remains constant over the bit time and thus just one time slot is required for the representation of a bit (other methods of bit encoding are e. g. Manchester or Pulse-width-modulation) (figure B.1).

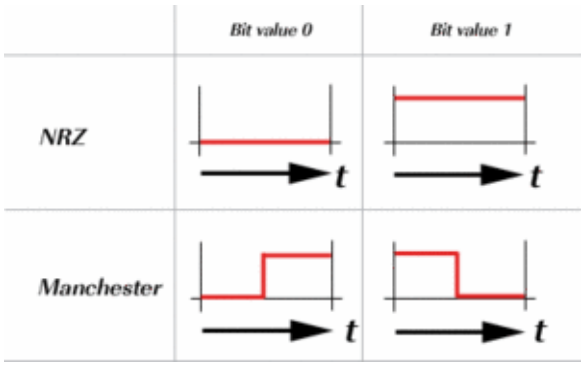


Fig. B.1: NRZ compared with Manchester bit representation

On the bit-level CAN uses synchronous bit transmission. This enhances the transmitting capacity but also means that a sophisticated method of bit synchronization is required. The CAN protocol regulates bus access by bit-wise arbitration. The signal propagation from sender to receiver and back to the sender must be completed within one bit-time. Bit stuffing is applied by inserting a complementary bit after five bits of equal value.

The physical media most commonly used to implement CAN networks is a differentially driven pair of wires with common return (figure B.2).

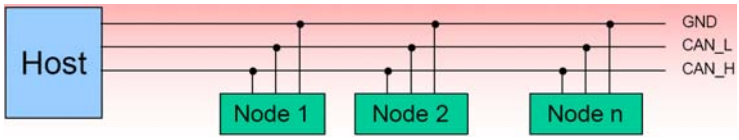


Fig. B.2: Physical CAN bus connection

The parameters of the electrical medium become important when the bus length is increased. Signal propagation, the line resistance and wire cross sections are factors when dimensioning a network. In order to achieve the highest possible bit rate at a given length, a high signal speed is required.

The devices of a humanoid robot that will be connected by a CAN network are sensors and intelligent motion control devices. A CAN message never reaches these devices directly, but instead a host-processor and a CAN Controller is needed between these devices and the bus.

Each node connected to the CAN network requires:

- CAN Controller (hardware with a synchronous clock). The basic tasks of a CAN controller are receiving and sending data bits. Receiving: the CAN Controller stores received bits (one by one) from the bus until an entire message is available that can then be obtained by the host processor (usually after the CAN Controller has triggered an interrupt). Sending: the host-processor stores its transmit-messages in a CAN Controller, which transmits the bits serially to the bus.
- Transceiver (possibly integrated into the CAN Controller). Its basic tasks are signal receiving and sending a physical level (voltage). Receiving: it adapts signal levels from the bus to levels that the CAN Controller expects and has protective circuitry that protects the CAN Controller. Sending: it converts the transmit-bit signal received from the CAN Controller into a signal that is sent onto the bus.
- Host-processor. The host-processor decides what received messages mean, and which messages it wants to transmit itself. For intelligent CAN devices the host processor is required. Sensors can also be connected to the host-processor.

B.3 LAN/WLAN

A local-area network (LAN) is a computer network covering a small area. Ethernet is now the most common bottom (data link layer) for LAN networks. Ethernet is a family of frame-based computer networking technologies for local area networks. The name comes from the physical concept of the ether. It defines a number of wiring and signalling standards for the physical layer.

Ethernet is standardized as IEEE 802.3. The combination of the twisted pair versions of Ethernet for connecting end systems to the network, along with the fibre optic versions for site backbones, is the most widespread wired LAN technology. Wi-Fi, the wireless LAN (WLAN) standardized by IEEE 802.11, is prevalent in small networks used for remote access and augmenting Ethernet in larger installations.

Overall, the benefits of using Ethernet in the motion control architecture for humanoids include:

- Lower cost devices, wiring, and junction box savings
- Common network connection for all device types
- Reduced proliferation of the number of networks to be supported
- Networking infrastructure better able to support future information exchange needs

Above the physical layer, Ethernet nodes communicate by sending each other data packets, blocks of data that are individually sent and delivered. As with other IEEE 802 LAN/WLAN, each Ethernet and Wi-Fi node is given a single 48-bit MAC address, which is used both to specify the destination and the source of each data packet. Network interface cards or chips normally do not accept packets addressed to other Ethernet stations. A data packet on the wire is called a frame. A frame viewed on the actual physical wire would show Preamble and Start Frame Delimiter, in addition to the other data. These are required by all physical hardware.

Due to the high speed data transmission of Ethernet, the ever-decreasing cost of the hardware needed to support it and the reduced space needed by twisted pair, Ethernet based networks are the preferable choice at *Organization* and *Control levels* of the architecture for motion control of humanoid robots. Finally, the availability of wireless version – IEEE 802.11 allows communication between a humanoid robot and HMI control system without using any cable, thus providing robot's autonomy.

Appendix C

CAN based communication protocols

APPENDIX C

CAN based communication protocols

C.1 CAN protocol

The CAN protocol laid down with relation to the physical CAN bus is an international standard defined in the ISO 11898. It comprises the Physical and Data Link layers of the 7-layer ISO/OSI reference model. CAN protocol was originally developed for use in automobiles for integrating all sensors into a high speed network. Therefore, the CAN protocol is functionally suited very well to sensorial data transmission. Thus, as mentioned above, the control architecture for humanoid robot motion control adopts CAN protocol for sensorial data transferring.

The data frame in the CAN protocol is the only frame for actual data transmission (Figure C.1). There are two message formats the CAN protocol supports. They are the base frame format with 11 identifier bits and extended frame format with 29 identifier bits. Although CAN controllers which support extended frame format messages are also able to send and receive messages in CAN base frame format, it is it is desirable to use a unique message format for all data nodes in the network.

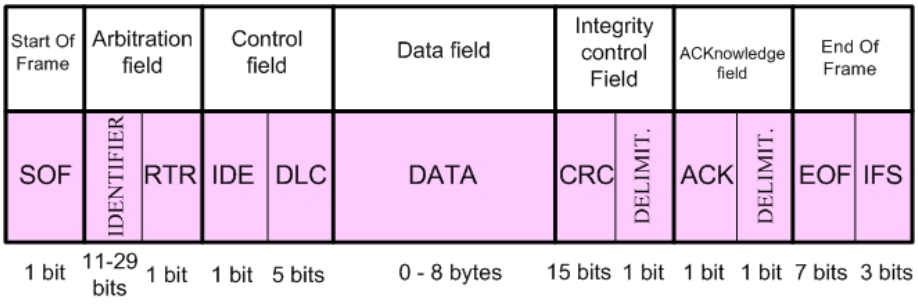


Fig. C.1: CAN message format

A CAN base frame message begins with a start bit called “Start Of Frame (SOF)”. This is followed by the “Arbitration field” which consists of the identifier and the “Remote Transmission Request (RTR)” bit used to distinguish between the data frame and the data request frame, called the remote frame. The subsequent “Control field” contains the “IDentifier Extension (IDE)” bit to distinguish between the CAN base frame and the CAN extended frame, as well as the “Data Length Code (DLC)” used to indicate the number of following data bytes in the “Data field”. If the message is used as a remote frame, the DLC contains the number of

requested data bytes. The “Data field” that follows is able to hold up to 8 data bytes. The integrity of the frame is guaranteed by the subsequent “Cyclic Redundant Check (CRC)” sum. The “ACKnowledge (ACK) field” compromises the ACK slot and the ACK delimiter. The bit in the ACK slot is sent as a recessive bit and is overwritten as a dominant bit by those receivers, which have at this time received the data correctly. Correct messages are acknowledged by the receivers regardless of the result of the acceptance test. The end of the message is indicated by “End Of Frame (EOF)”. The “Intermission Frame Space (IFS)” is the minimum number of bits separating consecutive messages. Unless another station starts transmitting, the bus remains idle after this.

C.2 CANopen protocol

The communication implemented on the bottom level between the main controller of a humanoid robot and the intelligent motion controllers of each joint, involves the integration of CANopen and the introduction of new functionality which is not contained within the standard CAN protocol used for the sensory data processing. Building CAN based communication systems guaranteeing interoperability and interchangeability of devices of different manufacturers requires a standardized application layer and profiles, standardizing the communication system, the device functionality and the system administration.

CANopen is the application layer which provides a set of services and protocols useful to every device on the network. The communication profile of CANopen provides the means to configure devices and communication data and defines how data is communicated between devices. Also, the CANopen specification includes the device and application profiles. These profiles define the rules of how specific devices (sensors, motion controllers, etc) from different suppliers can work together in the same CANopen network. In other words, a network with CANopen devices that complies with any device or application profile, provides pre-defined intercommunication between the devices.

The basic CANopen device and communication profiles are given in the CAN in Automation (CiA) standard DS 301. Profiles for more specialized devices are built on top of this basic profile, and are specified in numerous other standards released by CAN in Automation. CiA401 for I/O-modules and CiA402 for motion control profiles should be implemented in the motion control architecture for humanoid robots in order to provide the CAN based data transmission between intelligent motion controllers of humanoid joints.

Every device working into CANopen network has to implement certain standard features in its controlling software. It consists of three logical parts:

- Communication interface
- CANopen object dictionary
- Application interpreter software

Figure C.2 illustrates the CANopen device realization for the intelligent motion controller for a humanoid robot.

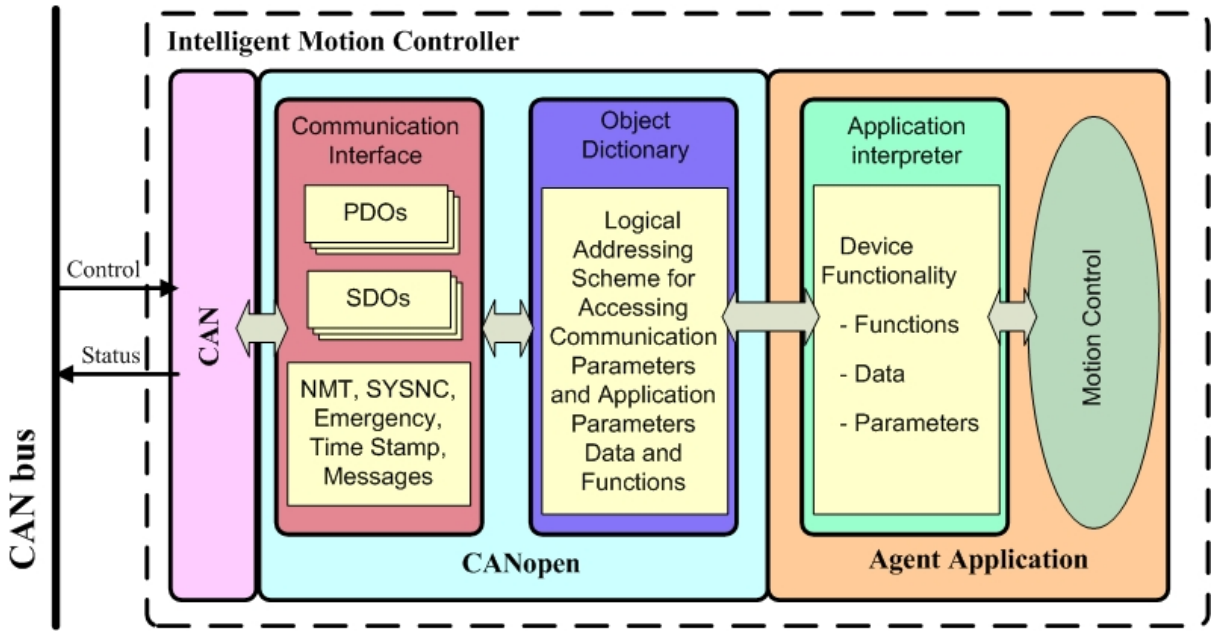


Fig. C.2: CANopen device realization for intelligent motion controller

The communication interface and protocol software handle the communication via the CAN network and implement the protocols for messaging with the other nodes in the network.

The CANopen object dictionary is an array of variables with a 16-bit index. It interfaces the protocol and the application software. It contains references for all used data types and stores all communication and application parameters.

The application software (motion control Agent) performs the desired function of the device (joint motion control in the case of intelligent motion controller). The application interpreter is configured by variables in the object dictionary and the data is sent and received through the communication interface.

Thus, implementing specific device profiles, every intelligent devices residing on the CANopen network can be realized and programmed. A system designer can configure and maintain a device via CANopen services due to the object dictionary definition.

Appendix D

Real-time data transmission

APPENDIX D

Real-time data transmission

D.1 Sensorial level Master-Slave real-time data transmission

In the master/slave relationship that is implemented between the Main controller and sensors of a humanoid robot, the control Server application is designated as the master, sending requests for data from the slaves (sensors). The real-time data transmission requirements can be fulfilled by specifying the priority for every sensorial data message. Thus, for example, control attitude data, more critical for humanoid robot stabilization (chapter 5), should have priority over force/torque data and therefore, be transmitted over CAN bus network with less delay.

The transmission priority of any CAN message compared to another is specified by the identifier field of each message. The priorities are laid down during communication system design in the form of corresponding binary values and cannot be changed dynamically. The identifier with the lowest binary number has the highest priority. Bus access conflicts are resolved by bit-wise arbitration of the identifiers involved, with each node observing the bus level bit for bit. This happens in accordance with the wired-and-mechanism, by which the dominant state overwrites the recessive state. All those nodes with recessive transmission and dominant observation lose the competition for bus access. In figure D1, node 1 loses the bus access trying to write to bit 5 of the identifier with recessive level and node 3 loses the access trying to write to the RTR bit. Therefore, node 2 wins the arbitration and sends its message to the network.

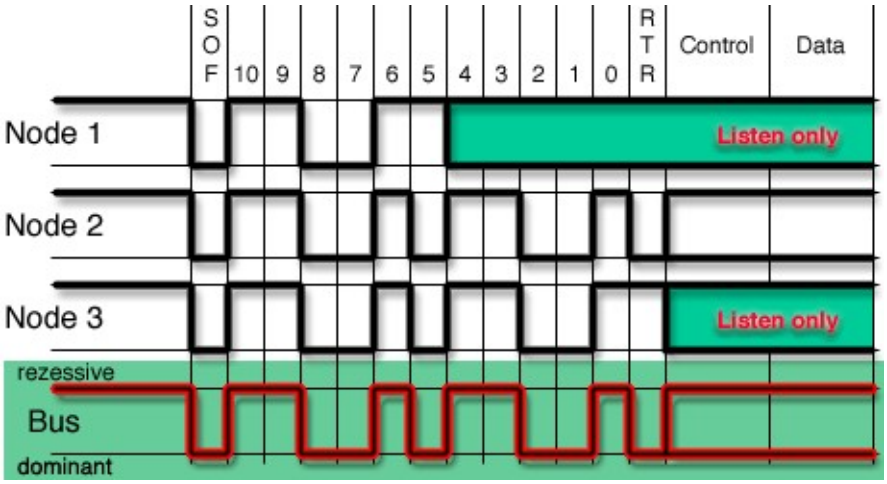


Fig. D1: CAN protocol real time arbitration

All those “losers” automatically become receivers of the message with the highest priority and do not re-attempt transmission until the bus is available again.

D.2 Device level Client-Server real-time data transmission

The client/server relationship is implemented between the Main controller and the intelligent motion controllers of joints of a humanoid robot. The Control Server application acts as a client sending data. Agent application acts as a server, which replies with one or more packages containing the requested data or acceptance confirmation. In real-time motion control systems of humanoid robots, the urgency of messages to be transmitted over the communication network differ. For example, target position points for motion controllers of a robot in the dynamic interaction mode (writing a sub-buffer into operational buffer) have to be transmitted with priority over other dimensions, e.g. polling for current position of a joint for visualization into an HMI system.

The real-time interaction can be realized by special handling of CANopen messages – the Dynamic object buffer. In dynamic object buffer mode the CAN messages and their data are stored in two object lists - transmission and reception lists (figure D.2).

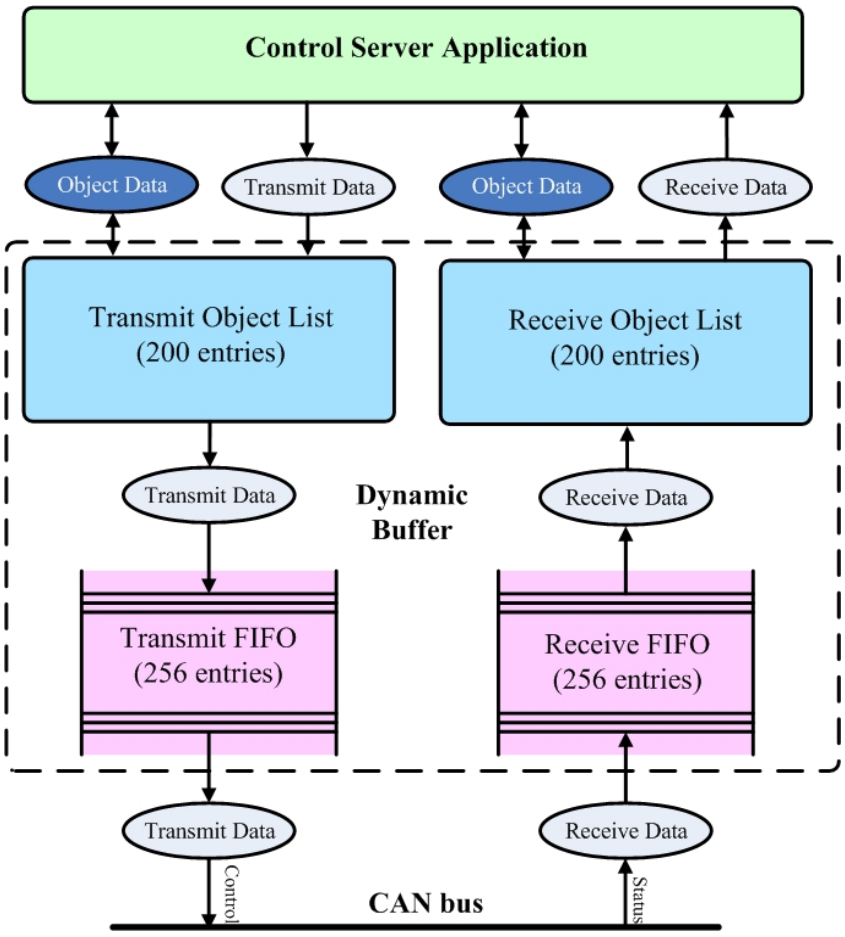


Fig. D.2: CANopen dynamic buffer

It is possible at any time to read or write the data of a defined object. Thus, the Control Server application always has a consistent representation of a defined “CANopen database”. The entries of the lists, i.e. CANopen messages of interest, have to be defined by the Control Server application. An object includes an identifier and data of a CANopen message. The application handles the objects with their object number. Thus, the objects with fewer object numbers have priority and will be transmitted (or received) first. The objects with upper object numbers will wait in the dynamic buffer until all upper level objects are transmitted (or received). In the proposed communication architecture a maximum of 200 transmission and receive objects may be defined. The handling of transmission requests, received messages, transmit acknowledges and remote frames can be individually switched on or off for each object by definition.

After an object is selected from the dynamic buffer to be transmitted, it is placed into the memory of the controller. The communication between the application and the CAN bus is processed sequentially using FIFOs (First In First Out). The message that is entered first into the FIFO , is the next to be processed further. Each of the FIFOs can bear a maximum of 256 entries.

In the case of receiving of an object from the CAN bus, it is received by the FIFO buffer and then similarly processed by a dynamic buffer to determine its priority and for sending to the Control Server application. The mechanism of dynamic buffering of messages into a CANopen network provides the real-time data transmission for designed communication infrastructure for humanoid robots.

Bibliography

Bibliography

[Akachi 2005] K. Akachi, K. Kaneko, N. Kanehira, S.Ota, G. Miyamori, M. Hirata, S. Kajita, and F. Kanehiro, Development of Humanoid Robot HRP-3P, Proceedings of 2005 5th IEEE-RAS International Conference on Humanoid Robots, 2005.

[Allen 99] J. Allen, Mixed initiative interaction. IEEE Intelligent Systems, 15(4):14–23, 1999.

[Arakawa 97] T. Arakawa and T. Fukuda, Natural motion of biped locomotion robot using hierarchical trajectory generation method consisting of GA, EP, layers, in Proc. IEEE Int. Conf. on Robotics and Automation ICRA97, pp. 211-216, New-Mexico, 1997.

[Arbulu 2005] M.Arbulu, F.Prieto, L.M.Cabas, P.Staroverov, D.Kaynov, C.Balaguer, ZMP Human Measure System, In Proc. of 8th International Conference on Climbing and Walking Robots (Clawar'2005), United Kingdom, 2005.

[Arkin 98] R.C. Arkin, Behaviour Based Robotics, MIT Press, Cambridge, 1998.

[Assif 98] Assif, D., Himel, R. & Grajower, Y. , A new electromechanical device to measure the accuracy of interocclusal records, Journal of Prosthetic Dentistry 59 (6), pp. 672-676, 1998.

[Astrom 94] K. J. Astrom and B. Wittenmark, Adaptive Control, Addison-Wesley, 2d ed. 1994.

[Atta-Konadu 2005] R. Atta-Konadu, S. Lang, C. Zhang and P. Orban, Design of a Robot Control Architecture, Proceedings of the IEEE International Conference on Mechatronics & Automation, 2005.

[Baerveldt 97] A.-J. Baerveldt and R. Klang, A Low-cost and Low-weight Attitude Estimation System for an Autonomous Helicopter, Proc. of IEEE Intl. Conf. on Intelligent Engineering Systems, pp. 391-395, 1997.

[Balaguer 2006] C. Balaguer talk in IURS 2006 Robotics Summer School on Humanoid Robots, Benicassim, 18-22/9/2006.

[Ballard 91] Ballard, D. H. (1991). Animate vision. Artificial Intelligence, 48(1):57–86.

[Barr 2007] M. Barr, Embedded Systems Glossary, Netrino Technical Library, 2007.

[Bass 2003] Bass, L., Clements, P., Kazman, R.: Software Architecture in Practice. Addison Wesley, London, UK (2003).

[Batarseh 2003] I. Batarseh, "Power Electronic Circuits" by John Wiley, 2003.

[Bic 2003] Bic, L. F., Shaw A. C., Operating Systems, Pearson: Prentice Hall, 2003.

[Blanco 2005] D.Blanco, C.Castejon, S.Kadhim, L.Moreno, Predesign of an Anthropomorphic Lightweight Manipulator. 8th Int. Conf. on Climbing and Walking Robots (CLAWAR2005), Sep, 2005.

[Bradley 91] D. Bradley et al, Mechatronics, Electronics in products and processes, Chapman and Hall Verlag, London, 1991.

[Brooks 86] Brooks, R. A.: A Robust Layered Control System for a Mobile Robot. IEEE Journal of Robotics and Automation, 1986, RA-2: 14-23.

[Brooks 91] R.A. Brooks, Intelligence without reason, In Proc. of the 8th Int. Joint Conf. on Artificial Intelligence, Sydney, Australia, pp. 569-595, 1991.

[Brunelli 2008] Brunelli, C., Garzia, F. & Nurmi, J., A coarse-grain reconfigurable architecture for multimedia applications featuring subword computation capabilities, Journal of Real-Time Image Processing 3 (1-2): 21-32, 2008.

[Carpenter 96] B. Carpenter, Architectural Principles of the Internet, RFC 1958, 1996.

[Dillmann 04] D.N. Ly, K. Regenstein, T. Asfour and R. Dillmann. A Modular and Distributed Embedded Control Architecture for Humanoid Robots. Proc. of IEEE/RSJ Intern. Conf. on Intelligent Robots and Systems (IROS 2004), Japan, Sep. 28 - Oct. 2, 2004.

[Dowling 95] K. Dowling, Robotics: comp. robotics Frequently Asked Questions" Available as a hypertext document at http://www.frc.ri.cmu.edu/robotics-faq. 90+ pages, 1995.

[Dugan 2003] R. Dugan, M. McGranaghan, S. Santoso, H. W. Beaty. Electrical Power Systems Quality. McGraw-Hill Companies, Inc.. ISBN 0-07-138622-X, 2003.

[Fiedler 97] Fiedler P. and Schilb C., "Open Architecture Systems for Robotic Workcells," IWACT 1997 Conference Proceedings, Columbia Ohio, 1997.

[Fink 78] D. G. Fink and H. W. Beaty, Standard Handbook for Electrical Engineers, Eleventh Edition,McGraw-Hill, New York, 1978.

[Fitzpatrick 2003] P. Fitzpatrick, Perception and perspective in robotics, Proceedings of the 25th Annual Conference of the Cognitive Science Society, Boston, 2003.

[Frampton 2003] Frampton, K.D., Martin, S.E. & Minor, K., The scaling of acoustic streaming for application in micro-fluidic devices, Applied Acoustics 64 (7): 681-692, 2003.

[Galliday 76] C. Galliday and H. Hemami, “Postural stability of the two-degreeof- freedom biped by general linear feedback,” IEEE Transactions on Automatic Control, vol. 21, no. 1, pp. 74–79, February 1976.

[Garlan 2000] D. Garlan, Software Architecture: a Roadmap, in The Future of Software Engineering, A. Finkekstein (Ed), ACM Press, 2000.

[GE 66] GE Company: Exoskeleton Prototype Project, Final Report on Phase I, Report S-67-1011, Schenectady, NY, 66.

[Gienger 2000] M. Gienger, K. Loffler, F. Pfeiffer, A Biped Walking and Running Robot, Proceedings of the 9th International Workshop on Robotics in Alpe-Adria-Danube Region (RAAD), pp. 23-28, 2000.

[Goswami 2004] A. Goswami and V. Kallem, Rate of change of angular momentum and balance maintenance of biped robots, Proc of IEEE Intl. Conf. on Robotics and Automation, vol. 4, pp. 3785-3790, 2004.

[Gottlieb 94] I.M. Gottlieb, Electric Motors & Control Techniques, 2nd Edition, TAB Books 1994.

[Grabbe 92] M.T. Grabbe, J.J. Carroll, D.W. Dawson, and Z. Qu. "Robust control of robot manipulators during constrained and unconstrained motion." In IEEE Transactions on Robotics and Automation, 1992, pp. 2146–2151.

[Hara 97] K. Hara, R. Yokokava and K. Sadao, Dynamic Control of Biped Locomotion Robot for Disturbance on Lateral Plane, Proc. of The Japan Society of Mechanical Engineers 72nd kansai meeting, pp.10_37-10_38, 1997 (in Japanese).

[Hashimoto 2005] K. Hashimoto, T. Hosobata, Y. Sugahara, Y. Mikuriya, H. Sunazuka, M. Kawase, H. Lim and A. Takanishi, Development of Foot System of Biped Walking Robot Capable of Maintaining Four-point Contact, Proceedings of the 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 1464-1469, 2005.

[Hemami 84] H. Hemami and B.-R. Chen, "Stability analysis and input design of a two-link planar biped," The International Journal of Robotics Research, vol. 3, no. 2, pp. 93–100, 1984.

[Hirai 98] K. Hirai, M. Hirose, Y. Haikawaand T. Takenaka, The development of Honda humanoid robot, in Proc. of IEEE Int. Conf. on Robotics and Automation ICRA98, pp. 1321-1326, Belgium, 1998.

[Hirukawa 2007] H. Hirukawa, S. Hattori, S. Kajita, K. Harada, K. Kaneko, F. Kanehiro, M. Morisawa and S. Nakaoka, "A Pattern Generator of Humanoid Robots Walking on a Rough Terrain", 2007 IEEE International Conference on Robotics and Automation Roma, Italy, April 2007.

[Hornby 2000] Hornby, G.S. Takamura, S. Yokono, J. Hanagata, O. Yamamoto, T. Fujita, M, Evolving robust gaits with AIBO, Proceedings of IEEE International Conference on Robotics and Automation, ICRA'00, vol.3, pp. 3040-3045, 2000.

[Huang 2001] Huang, Q., Yokoi, K., Kajita, S., Kaneko, K., Arai, H., Koyachi, N., Tanie, K., "Planning Walking Patterns for a Biped Robot," IEEE Trans. on Robotics and Automation, Vo1.17, No.3, June 2001.

[Jardon 2006] A. Jardon Huete, Metodologia de diseno de robots asistenciales. Aplicacion al robot portatil ASIBOT. , Ph.D. Thesis, 2006.

[Kagami 2000] S.Kagami, K.Nishiwaki, J.J.Kuffner, T.Sugihara, M.Inaba, H.Inoue, Design and Implementation of Humanoid H6 and its Application to Remote Operation,Proc. of 7th Int. Symp. on Exprimental Robotics, 2000.

[Kagami 2001] S. Kagami, K. Nishiwaki, J.J. Kuffner, K. Okada, M. Inaba, H. Inoue, Low-level Autonomy of Remote Operated Humanoid Robot H6 & H7, Preprint of 10th International Symposium of Robotics Research, 2001.

[Kaiser 2001] D. Kaiser, Fundamentals of Servo Motion Control, Parker Compumotor whitepaper, 2001.

[Kajita 01] S. Kajita, K. Yokoi, M. Saigo and K. Tanie, Balancing a Humanoid Robot Using Backdrive Concerned Torque Control and Direct Angular Momentum Feedback, Proceedings of the 2001 IEEE International Conference on Robotics & Automation, 2001.

[Kajita 2001] S. Kajita, F. Kanehiro, K. Kaneko, K. Yokoi and H. Hirukawa, The 3D Linear Inverted Pendulum Mode: A simple modeling for a biped walking pattern generation. In Proc. IEEE Conf on Intelligent Robots and Systems (IROS 2001), pp. 239-246, Hawaii, 2001.

[Kajita 2003] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi and H. Hirukawa, Biped Walking Pattern Generation by using Preview Control of Zero-Moment Point, in Proc. of IEEE Int. Conf. on Robotics and Automation (ICRA2003), pp. 1620-1626, Taiwan, 2003.

[Kajita 91] S. Kajita and K. Tani, Study of Dynamic Biped Locomotion on Rugged Terrain, in Proc. of IEEE Int. Conf. on Robotics and Automation (ICRA1991), pp. 1405-1410, 1991.

[Kajita ICRA01] S.Kajita, O. Matsumoto and M. Saigo, Real-time 3D walking pattern generation for a biped robot with telescopic legs, in Proc. of IEEE Int. Conf. on Robotics and Automation (ICRA2001), pp. 2299-2306, 2001.

[Kanehiro 99] F. Kanehiro, Y. Tamiya, M. Inaba, H. Inoue, "Developmental Methodology for Building Whole Body Humanoid System", Proc. of the IEEE/RSJ International Conference on Intelligent Robotics and Systems (IROS'99), Kyongju Korea, pp.1210-1215, 1999.

[Kaneko 2002] K. Kaneko, F. Kanehiro, S. Kajita, K. Yokoyama, K. Akachi, T. Kawasaki, S. Ota and T. Isozumi, Design of Prototype Humanoid Robotics Platform for HRP, Proc. of IEEE/RSJ Int. Conference on Intelligent Robots and Systems, pp. 2431-2436, 2002.

[Kaneko 2004] K. Kaneko, F. Kanehiro, S. Kajita, H. Hirukawa, T. Kawasaki, M. Hirata, K. Akachi, and T. Isozumi, Humanoid Robot HRP-2, Proceedings of the 2004 IEEE International Conference on Robotics & Automation, New Orleans, USA, 2004.

[Kee 2004] D. Kee, G. Wyeth, J. Roberts, “Biologically Inspired Joint Control for a Humanoid Robot”, Proc. of 4th IEEE/RAS International Conference on Humanoid Robots, 2004, Vol. 1, pp. 385- 401.

[Khamis 2007] A.M.Khamis; M.S.Kamel; M.A.Salichs. Human-Robot Interfaces for Social Interaction. International Journal of Robotics and Automation. Vol. 22. No. 3. pp.215-221. 2007.

[Kim 2002] J. Kim, S. Park, I. Park and J. Oh, Development of a Humanoid Biped Walking Robot Platform KHR-1 - Initial Design and Its Performance Evaluation, Proceedings of The Third IARP International Workshop on Humanoid and Human Friendly Robotics Tsykyba, Japan, P. 14 - 21, 2002.

[Kim 2004] J. Kim and J. Oh, Walking Control of the Humanoid Platform KHR-1 based on Torque Feedback Control, Proc. of IEEE Intl. Conf. on Robotics and Automation, pp. 623-628, 2004.

[Kramer 93] J. Kramer, Open Software Architecture for the Semiconductor Industry, Proceedings of IEEE/SEMI Advanced Semiconductor Manufacturing Conference and Workshop, 1993.

[Kretschmar 2005] M. Kretschmar and S. Welsby, Capacitive and Inductive Displacement Sensors, in Sensor Technology Handbook, J. Wilson editor, Newnes: Burlington, MA, 2005.

[Kuo 95] A. Kuo, "An optimal control model for analyzing human postural balance," IEEE Transactions on Biomedical Engineering, vol. 42, no. 1, pp. 87–101, January 1995.

[Lakie 2003] M. Lakie, N. Caplan and I. D. Loram, Human balancing of an inverted pendulum with a compliant linkage: neural control by anticipatory intermittent bias, The Journal of Physiology, pp. 357-370, 2003.

[Loffler 2004] K. Loffler, M. Gienger, F. Pfeiffer, Sensors and Control Concept of a Biped Robot, IEEE Transactions on Industrial Electronics, vol. 51 n5,October 2004.

[Lewin 2001] C. Lewin, Distributed Motion Control: A Worthy Option for Connectivity, Performance Motion Devices Inc., Control Engineering , 2001.

[Lewis 2003] F. Lewis, D. Dawson, C.T. Abdallah, "Robot Manipulator Control: Theory and Practice", 2nd ed., CRC, 2003.

[Lim 2000] H. Lim, A. Takanishi, Waseda Biped Humanoid Robots Realizing Human-like Motion, Proceedings of the International Workshop on Advanced Motion Contorol, pp. 525-530, 2000.

[Lu 2007] J. Lu, Multi-objective Group Decision Making: Methods, Software and Applications With Fuzzy Set Techniques, London : Imperial College Press, cop. 2007.

[MacDorman 2005] K. F. MacDorman, T. Minato, M. Shimada, S. Itakura, S. Cowley, and H. Ishiguro, Assessing Human Likeness by Eye Contact in an Android Testbed, Proceedings of XXVII Annual Conference of the Cognitive Science Society CogSci2005, 2005.

[Matijevics 2007] I. Matijevics, Infrared Sensors Microcontroller Interface System fof Mobile Robots, In Proceedings of the 5th International Symposium on Intelligent Systems and Informatics, 2007.

[Maybeck 99] P. S. Maybeck, Stochastic models estimation and control vol. 1, Academic Press Inc., 1999.

[McGeer 90] T. McGeer. Passive Dynamic Walking. International Journal of Robotics Research. Vol. 9, pp. 62-82, 1990.

[Meyers 97] Meyers, B. Craig, and Obemdorf, Patricia A. Open Systems: The Promises and the Pitfalls (course). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1997.

[Nakamura 99] A. Dasgupta and Y. Nakamura. Making feasible walking motion of humanoids robots from human motion capture data, in Proc. of the IEEE Int. Conference on Robotics and Automation ICRA99, pp. 1044-1049, Detroit, 1999.

[Nwana 96] H. S. Nwana., Software Agents: An Overview, Knowledge Engineering Review, Cambridge University Press, 1996.

[Oberndorf 97] P. A. Oberndorf, Facilitating Component-Based Software Engineering: COTS and Open Systems, In Proceedings of the 5th International Symposium on Assessment of Software Tools (SAST '97) , p. 0143, 1997.

[Ogura 2006] Y. Ogura, H. Aikawa, K. Shimomura, H. Kondo, A. Morishima, H. Lim and A. Takanishi, Development of A Humanoid Robot WABIAN-2, Proceedings of the 2006 IEEE International Conference on Robotics and Automation, pp.76-81, 2006

[Ollero 2001] A. Ollero Baturone, ROBOTICA. Manipuladores y robots moviles, Marcombo, S.A., 2001.

[Ollero 94] A. Ollero, A. Mandow, J. Gomez de Gabriel, Control architecture for mobile robot operation and navigation, Robotics and Computer-Integrated Manufacturing. Vol. 11, n. 4, pp.259-269, Pergamon.

[Park 2005] I. Park, J. Kim, S. Park, and J. Oh, Development of Humanoid Robot Platform KHR-2 (KAIST Humanoid Robot-2). International Journal of Humanoid Robotics, Vol. 2, No. 4, 2005, pp. 519 - 535.

[Park 2007] I. Park, J. Kim, J. Lee, and J. Oh, Mechanical Design of the Humanoid Robot Platform, HUBO, Advanced Robotics, Vol. 21, No. 11, 2007.

[Perens 99] B. Perens., Open Sources: Voices from the Open Source Revolution, January 1999, ISBN 1-56592-582-3.

[Perry 2002] D. Perry, Force/torque sensors help industrial robots make the right moves, ATI Industrial Automation, 2002.

[Pratt 96], J. Pratt, A. Torres, P. Dilworth, G. Pratt, Virtual Actuator Control, Preceedings of IROS '96, Osaka, Japan, 1996.

[Proctor 2003] F. M. Proctor, J. L. Michaloski, Overview of Communication Standards relating to Motion Control Systems, National Institute of Standards and Technology.

[Quinn 98] R. D. Quinn and R. E. Ritzmann, Construction of a Hexapod Robot with Cockroach Kinematics Benefits both Robotics and Biology, Connection Science, Vol. 10, Nos 3&4, pp. 239-254, 1998.

[Raymond 2004] E. S. Raymond, R.W. Landley, The Art of Unix Usability, On-line book http://www.catb.org/~esr/writings/taouu/html/, 2004.

[Raymond 99] E. S. Raymond, The Cathedral & the Bazaar. O'Reilly, 1999, ISBN 1-56592-724-9.

[Regenstein 2003] K. Regenstein and R. Dillmann, "Design of an open hardware architecture for the humanoid robot ARMAR," in Proc. of IEEE International Conference on Humanoid Robots (Humanoids 2003), Karlsruhe and Munich, Germany, 2003.

[RFC 1122] R. Braden, Editor, RFC 1122, Requirements for Internet Hosts -- Communication Layers , 1989.

[Robinson 99] D. W. Robinson, J. E. Pratt, D. J. Paluska, G. A. Pratt, Series Elastic Actuator Development for a Biomimetic Walking Robot, Proceedings of the IEEE/ASME International Conference on Advanced Intelligent Mechatronics, 1999

[Robles 2004] Robles, G. (2004). "A Software Engineering approach to Libre Software", in Robert A. Gehring, Bernd Lutterbeck: Open Source Jahrbuch 2004, Berlin: Lehmanns Media.

[Sadoski 97] Sadoski D., Client/Server Software Architectures--An Overview, Software Technology Roadmap, 1997.

[Sage 90] A. P. Sage, editor: Concise Enchyclopedia of Information Processing in Systems and Organizations, Pergamon, New York, 1990.

[Sakagami 2002] Y. Sakagami, R. Watanabe, C. Aoyama, S. Matsunaga, N. Hikagi and K. Fujimura, "The intelligent ASIMO: system overview and integration", Proc. of IEEE/RSJ Int. Conference on Intelligent Robots and Systems, pp. 2478-2483, 2002.

[Schraft 94] Schraft R.D., Mechatronics and robotics for service applications, IEEE Robotics Automation Magazine, Vol. 1, No 4, December 1994.

[Seaman 2004] G. Seaman, Techonology and Society, 3rd Oekonux conference "Wealth by copyleft -- Creativity in the digital age", Vienna University, Austria, 2004.

[Simon 2006] D.Simon, Optimal State Estimation: Kalman, H-infinity, and Nonlinear Approaches, John Wiley & Sons, 2006

[Srinivasan 2007] Srinivasan, K. Gu, J., Multiple Sensor Fusion in Mobile Robot Localization, Procedings of the Canadian Conference on Electrical and Computer Engineering, pp. 1207-1210, 2007.

[Stevens 94] W. Richard Stevens, TCP/IP Illustrated, Volume 1: The Protocols, Addison-Wesley, ISBN 0-201-63346-9, 1994

[Sugahara 2004] Y. Sugahara, T. Endo, T. Hosobata, Y. Mikuriya, H. Lim and A. Takanishi, WL-15: Prototype of a Multi-purpose Biped Locomotor with Parallel Mechanism, Proceedings of the 15th CISM-IFToMM Symposium on Robot Design, Dynamics and Control, 2004.

[Tanner 81] William R. Tanner, Industrial Robots, Robotics International of SME, Society of Manufacturing Engineers, Marketing Services Dept., 1981.

[Torre 2004] S. de Torre, L. M. Cabas, M. Arbulu, C. Balaguer Inverse Dynamics of Humanoid Robot by Balanced Mass Distribution Method, Proceedings 2004 IEEE-RSJ international Conference on Intelligent Robots and Systems, 2004.

[VanDoren 2003] V. J. VanDoren, Techniques for Adaptive Control, Elsevier, 2003.

[Verts 2008] Verts, William T. "Open source software." World Book Online Reference Center, 2008.

[Vertut 85] J. Vertut, P. Coiffet, Teleoperation and Robotics, Springer, ISBN 0850385881, 1985.

[Veruggio 2007] G. Veruggio, EURON Roboethics Roadmap, Release 1.2 (January 2007).

[Vukobratovic 2004] M. Vukobratovic, B. Borovac, Zero-Moment Point – thirty five years of its life, Internation Journal of Humanoid Robotics, Vol. 1, No.1, pp. 157-173, World Scientific , 2004.

[Vukobratovic 69] Vukobratovic M., Juricic D.: Contribution to the Synthesis of Biped Gait, IEEE Transactions on Biomedical Engineering, Vol. 16, No. 1, 1969.

[Vukobratovic 70] M. Vukobratovic, A. A. Frank, and D. Juricic, "On the stability of biped locomotion," IEEE Transactions on Biomedical Engineering, pp. 25–36, January 1970.

[Welch 2004] G. Welch and G. Bishop, An introduction to the Kalman filter TR 95-041, Department of Computer Science University of North Carolina, Chapel Hill, 2004.

[Westervelt 2003] Eric R. Westervelt, Toward a Coherent Framework for the Control of Planar Biped Locomotion, Ph.D. dissertation, Dept. Elect. Eng., Univ. of Michigan, 2003.

[Wu 85] C. Wu, “Compliance Control of a Robot Manipulator Based on Joint Torque Servo”, The International Journal of Robotics Research, 1985, Vol. 4, No. 3, 55-71.

[Xie 03] M. Xie: Fundamentals of Robotics. Linking perception to action, World Scientific, 2003.

[Yamaguchi 99] Yamaguchi, J., Soga, E., Inoue, S. and Takanishi, A., "Development of a Bipedal Humanoid Robot – Control Method of Whole Body Cooperative Dynamic Biped Walking -," Proc. of the 1999 ICRA, pp.368- 374, 1999.

[Yang 2005] P. Yang, X. Kong, Z. Liu, H. Chen, Q. Zhao, J. Liu, “Hybrid Intelligent Joint Controller for Humanoid Robot”, Proc. of Fourth International Conference on Machine Learning and Cybernetics, 2005.

[Yim 2000] Yim, M., D.G. Duff, K.D. Roufas. PolyBot: A Modular Reconfigurable Robot. in Proc. of the IEEE Int. Conference on Robotics and Automation. 2000.

[Zhang 2006] H. Zhang, W. Wang, Z. Deng, G. Zong & J. Zhang, A Novel Reconfigurable Robot for Urban Search and Rescue, International Journal of Advanced Robotic Systems, vol 3, pp. 359-366, 2006.

[Ziegler 42] Ziegler, J.G., and Nichols, N.B., Optimum Settings for Automatic Controllers, Transactions of the American Society of Mechanical Engineers (ASME). v. 64, 1942, pgs. 759-768.

[Zimmerman 2008] Zimmermann, M., Volden, T., Kirstein, K.-U., Hafizovic, S., Lichtenberg, J., Brand, O. & Hierlemann, A., A CMOS-based integrated-system architecture for a static cantilever array, Sensors and Actuators B: Chemical 131 (1), pp.254-264, 2008.

Related Publications

Related Publications

The main preliminary contributions of this thesis have already been published as:

- D.Kaynov; C.Balaguer. Joint Control of a Humanoid Robot. IEEE-RAS International Conference on Humanoid Robots (HUMANOIDS 2007). Pittsburgh. USA. Nov, 2007.
- M.Arbulú; L.M.Cabas; D.Kaynov; P.Staroverov; C.Balaguer. Trends of new robotics platform, designing Humanoid Robot Rh-1. CARS & FOF 07 23rd ISPE International Conference on CAD/CAM Robotics and Factories of the Future. Bogota. Colombia. Aug, 2007.
- M.Arbulú; D.Kaynov; L.M.Cabas; P.Staroverov; C.Balaguer. Nuevas tendencias en plataformas de robótica, caso robot humanoide Rh-1. Intercon 2007 XIV Congreso Internacional de Ingeniería Eléctrica, Electrónica y Sistemas. Piura. Peru. Aug, 2007.
- M.Arbulú; L.M.Cabas; P.Staroverov; D.Kaynov; C.Balaguer. Foot planning motion of humanoid robot Rh-1 using Lag algorithm. Clawar 2007. Singapore. Singapore. Jul, 2007.
- D.Kaynov; M.Arbulú; P.Staroverov; L.M.Cabas; C.Balaguer. Software and communication infrastructure design of the Humanoid. Clawar 2007. Singapore. Singapore. Jul, 2007.
- P.Staroverov; D.Kaynov; M.Arbulú; L.M.Cabas; C.Balaguer. Detecting sound sources with the humanoid robot Rh-1. Clawar 2007 . Singapore. Singapore. Jul, 2007.
- P.Staroverov; D.Kaynov; M.Arbulú; L.M.Cabas; C.Balaguer. Creating a gesture recognition system based on shirt shapes. Clawar 2007. Singapore. Singapore. Jul, 2007.
- D.Kaynov; C.Balaguer. Industrial automation based approach to design control system of the humanoid robot. IEEE International Symposium on Industrial Electronics ISIE 2007. Vigo. Spain. Jun, 2007.
- D.Kaynov; M.Arbulú; C.Balaguer. Arquitectura de control para la marcha dinámica de los robots humanoides. Aplicación al robot Rh-1. Workshop. Arquitecturas de control para robots. Madrid. Spain. Feb, 2007.
- L.M.Cabas; R.Cabas; P.Staroverov; M.Arbulú; D.Kaynov; C.Pérez; C.Balaguer. Mechanical Calculations on a Humanoid Robot. 9th International Conference on Climbing and Walking Robots (Clawar 2006). Brussels. Belgium. Sep, 2006.
- L.M.Cabas; R.Cabas; P.Staroverov; M.Arbulú; D.Kaynov; C.Pérez; C.Balaguer. Challenges in the design of the humanoid robot RH-1. 9th International Conference on Climbing and Walking Robots (Clawar 2006). Brussels. Belgium. Sep, 2006.
- M.Arbulú; L.M.Cabas; P.Staroverov; D.Kaynov; C.Pérez; C.Balaguer. On-line walking patterns generation for RH-1 Humanoid Robot using a simple three-dimensional inverted pendulum model. 9th International Conference on Climbing and Walking Robots (Clawar 2006). Brussels. Belgium. Sep, 2006.

- P.Staroverov; M.Arbulú; L.M.Cabas; D.Kaynov; C.Pérez; C.Balaguer. A Voice Controlled Image Recognition System. 9th International Conference on Climbing and Walking Robots (Clawar 2006). Brussels. Belgium. Sep, 2006.
- D.Kaynov; M.Arbulú; L.M.Cabas; P.Staroverov; C.Pérez; C.Balaguer. Control Architecture for the dynamic humanoid robot walking. Application to the RH-1 robot. 9th Internacional Conference on Climbing and Walking Robots (Clawar 2006). Brussels. Belgium. Sep, 2006.
- M.Arbulú; J.M.Pardos; L.M.Cabas; P.Staroverov; D.Kaynov; C.Pérez; M.A.Rodríguez; C.Balaguer. Rh-0 humanoid full size robot`s control strategy based on the Lie logic technique. IEEE-RAS International Conference on Humanoid Robots (Humanoids'2005). Tsukuba. Japan. Dec, 2005.
- D.Kaynov; M.A.Rodríguez; M.Arbulú; P.Staroverov; L.M.Cabas; C.Balaguer. Advanced motion control system for the humanoid robot Rh-0. 8th International Conference on Climbing and Walking Robots (Clawar 2005). London. United Kingdom. Sep, 2005.
- P.Staroverov; C.Chicharro; D.Kaynov; M.Arbulú; L.M.Cabas; C.Balaguer. “T-Shirt Based” Image Recognition System. 8th International Conference on Climbing and Walking Robots (Clawar 2005). London. United Kingdom. Sep, 2005.
- L.M.Cabas; S.Torre; R.Cabas; D.Kaynov; M.Arbulú; P.Staroverov; C.Balaguer. Mechanical design and dynamic analysis of the humanoid robot Rh-0. 8th International Conference on Climbing and Walking Robots (Clawar 2005). London. United Kingdom. Sep, 2005.
- M.Arbulú; F.Prieto; L.M.Cabas; P.Staroverov; D.Kaynov; C.Balaguer. ZMP Human Measure System . 8th International Conference on Climbing and Walking Robots (Clawar'2005). London. United Kingdom. Sep, 2005.